Real-Time Spatio-Temporal Query Processing in Mobile Ad-Hoc Sensor Networks

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

Technical Report BUCS-TR-2006-028
October 15, 2006

***Abstract*— Personal communication devices are increasingly equipped with sensors that are able to collect and locally store information from their environs. The mobility of users carrying such devices, and hence the mobility of sensor readings in space and time, opens new horizons for interesting applications. In particular, we envision a system in which the collective sensing, storage and communication resources, and mobility of these devices could be leveraged to query the state of (possibly remote) neighborhoods. Such queries would have spatio-temporal constraints which must be met for the query answers to be useful. Using a simplified mobility model, we analytically quantify the benefits from cooperation (in terms of the system's ability to satisfy spatio-temporal constraints), which we show to go beyond simple space-time tradeoffs. In managing the limited storage resources of such cooperative systems, the goal should be to minimize the number of unsatisfiable spatio-temporal constraints. We show that Data Centric Storage (DCS), or "directed placement", is a viable approach for achieving this goal, but only when the underlying network is well connected. Alternatively, we propose, "amorphous placement", in which sensory samples are cached locally, and shuffling of cached samples is used to diffuse the sensory data throughout the whole network. We evaluate conditions under which directed versus amorphous placement strategies would be more efficient. These results lead us to propose a hybrid placement strategy, in which the spatio-temporal constraints associated with a sensory data type determine the most appropriate placement strategy for that data type. We perform an extensive simulation study to evaluate the performance of directed, amorphous, and hybrid placement protocols when applied to queries that are subject to timing constraints. Our results show that, directed placement is better for queries with moderately tight deadlines, whereas amorphous placement is better for queries with looser deadlines, and that under most operational conditions, the hybrid technique gives the best compromise.**

I. 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 [1], [17]. While sensors may be embedded into mobile devices in support of local applications, the communication and storage 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 spatiotemporal sensory data, which may prove quite useful – *e.g.*, allowing soldiers in a battlefield to query conditions at remote locations without relying on backend systems, or 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.

An interesting motivating application comes from the military field. Military research labs are designing and producing new wearable units to enable soldiers to cover wider areas of the battlefield while maintaining a high level of efficiency in communication and maneuvering [2], [3]. Each soldier is equipped with a backpack that has multiple sensors (*e.g.*, motion sensors, acoustic sensors, infrared light emitting diodes, and pan/tilt cameras) with the goal of programming these units remotely to perform certain tracking or monitoring tasks, while the soldiers roam the field. Samples gathered by these sensors may prove very useful to other soldiers in the field. Imagine a scenario in which a soldier is interested in moving to a certain location; information about any movement detected in that location during the last five minutes would be a crucial piece of knowledge. Moreover, providing this information to that soldier *on time* will be even more important. In case a group of soldiers are temporarily out of communication with their base station, the task of communicating this information to the interested soldier becomes a distributed challenge to which the system has to respond.

Prior research in which sensor networks are viewed as "databases" that may be used to store sensory data and to answer queries thereupon was mostly concerned with issues of efficient representation (*e.g.*, [7]), aggregation and summarization (*e.g.*, [5], [18]), and routing (*e.g.*, [10]).

Embedding sensing abilities into mobile devices allows us to remove a number of key assumptions often made in this prior research. First, prior studies have mostly been concerned with fields in which sensors are densely deployed so as to leverage the limited-range radio communication abilities of the sensors in setting up the "network". In particular, in these studies, there is an inherent assumption that once disconnected

§ This work was supported in part by a number of NSF awards, including CNS Cybertrust Award #0524477, CNS NeTS Award #0520166, CNS ITR Award #0205294, and EIA RI Award #0202067.

from the network, a sensor node is not useful since there is no way to access the sensory data collected by that node from any other part of the network. Clearly, this is not the case if sensor nodes are mobile, since it is possible for such nodes to become reconnected by virtue of their or other nodes' mobility. This also implies that in the presence of mobility, good spatio-temporal coverage of a field is possible with a sparse population of sensors. Second, prior studies have mostly been concerned with resource-impoverished sensor nodes, with an added emphasis on preserving battery power and on the efficient use of very limited memory as exemplified in [9]. The integration of sensors into mobile devices allows us to loosen these constraints a bit since these devices are likely to possess richer computing and storage resources, and can allow for power restoration once a node moves to a location in which the battery could be recharged. Third, the availability of unused, spare resources — memory in particular — makes it possible for such devices to be used as vehicles for ferrying data between two disconnected sensor network neighborhoods, when the device in which this memory resides moves from one neighborhood to the other.

The above arguments suggest that the mobility of sensor nodes could be leveraged to improve (possibly significantly) the query performance in such networks, especially when the sensor field is sparsely populated. Prior research (reviewed in Section IX) leveraged nodal mobility and (typically large) *persistent* storage to improve the delivery rate of non-real-time queries in an *on-demand* fashion, *i.e.*, when a query arrives, it is opportunistically routed from node to node until it reaches a node which has the answer. The answer is then routed back to the inquirer in the same opportunistic manner.

Our Contribution: In this paper, we leverage the *mobility* and the *limited storage* in a set of nodes to improve the recall rates for local queries over the field in which the nodes are roaming. Queries we consider have spatio-temporal constraints. Each query targets a specific physical location — *e.g.*, what MAC addresses were seen at location X. Since having an observer 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. This is an example of a *spatial constraint*. Additionally, each query may specify recency constraints for observations to be considered valid, or deadlines for the results to be returned. These would be examples of *temporal constraints*.

The local cache at a node in the system could be seen as storing 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 the physical/geographical location in which the sample was taken as well as the time in which the sample was taken. Clearly, the cache memory in a sensor node must be managed in a manner that maximizes its utility. 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.

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" to refer to the former of these approaches and the term "amorphous placement" to refer to the latter.

Directed placement has been proposed and evaluated in a number of studies in peer-to-peer networks [27], [24] as well as in sensor networks [4], [23], [25]. When used in conjunction with sensor databases, directed placement has been termed as Data Centric Storage (DCS) [26]. In our context of spatio-temporal data, using directed placement, 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 spatiotemporal subspace to which this sample belongs. Such transport could be carried using any number of multi-hop ad-hoc routing techniques [12], [13], [22]. Directed placement simplifies query processing significantly, since a well-defined "home" for a spatio-temporal subspace makes it straightforward to route future queries over that space. Moreover, partitioning the spatio-temporal space over the various nodes in the system allows the system to collectively store a larger number of samples, since ideally, a reading is stored only in a single entry at its "home" node or set of nodes. For these reasons, in ad-hoc or sensor networks, directed placement is expected to work well, but only when the connectivity of the underlying network is rich enough to support the transport/routing of samples/queries from one point in the network to another (the "home") over multi-hop routes. In sparse, often partitioned networks such as those we envision in this paper, directed placement may not perform well.

Instead, in this paper, we propose the use of amorphous placement, whereby a reading is not associated with a locale to which it must move, 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 exchange a small number of samples. This exchange "diffuses" samples from different spots in the field to the rest of the nodes that may have never visited these spots. Clearly, this approach requires that some form of constrained flooding be used to locate samples belonging to a spatio-temporal subspace of interest (*i.e.*, answers to any given query). In dense, well-connected networks, flooding techniques are viewed as wasteful of bandwidth and power, but as we show in this paper, in sparse, often partitioned networks, the combination of associative (and possibly replicated) placement of samples coupled with *limited* flooding-based routing (even just one-hop broadcast) of queries tend to improve the recall rates of spatially-constrained and temporally-constrained queries.

Directed and amorphous placement approaches represent two extremes. In this paper, we also consider a hybrid approach, in which the spatio-temporal constraints associated with queries of a particular sensory data type determine the most appropriate placement strategy for that data type. This

determination is clearly a function of many system parameters (*e.g.*, connectedness of underlying ad-hoc network, size of available cache, *etc.*).

Paper Outline: In Section II we define and motivate our query and data models. In Section III, we show that cooperation between mobile nodes allows the system to meet more stringent deadlines compared to the case with no cooperation. In Section IV, we build on this finding by providing the design space (and details) of the two basic approaches that allow us to bring cooperation to bear – namely, Amorphous Placement and Retrieval (APR) and Directed Placement and Retrieval (DPR). In Section V, we present an experimental setting that allows us to contrast in Section VI the suitability of APR and DPR for handling spatio-temporal constraints at different timescales. In Section VII, we present the Hybrid Placement and Retrieval (HPR) protocol that allows data sets associated with queries in different timescales to be managed (adaptively) by either APR or DPR, as appropriate. In Section VIII, we present results from extensive simulations that we used to evaluate the performance of DPR, APR, and HPR under various conditions and for workloads reflecting different mixes of timing constraints. We place our work in perspective and discuss related work in Section IX. We conclude with a summary and an overview of on-going work in Section X.

II. Query and Data Models

Query Precision: One particularly important parameter of queries over a given location or region is the tolerable inaccuracy that the query allows in the result. We assume that queries target a specific location in the field along with some desirable *precision* (ℓ), which constrains how far the readings used to answer the query could be from the actual location of the query target. Notice that introducing query precision allows us to deal with applications in which queries might target locations in the field where no samples were ever collected.

Query Deadline: Another parameter of queries is the deadline, if a node is able to obtain an answer within tolerable precision *by the query deadline*, then the query is counted as a success, otherwise, it is deemed to have missed its deadline – a failure. In this paper, and unless otherwise specified, we assume that all queries have the same query precision and deadline.

Query Target: Different applications may exhibit different distributions of query targets relative to the query origin/sink (the location of the node submitting the query). Clearly, one can think of many applications that exhibit a correlation between query origins and targets. Such correlations, if known *a priori*, 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) [16]. The consideration of such issues is application-specific (since one must justify the specific aspects of the workload), and is beyond the scope of our work. Thus, in this paper, we assume that nodes are equally interested in the whole field, *i.e.*, there is no correlation betwen a query origin and its target, which is selected uniformly at random over the entire field[1].

Data Freshness: In order to be useful, the returned answer to a query should not be "stale", *i.e.*, it should be collected relatively recently. This guarantees that each query answer is an accurate representation of the current state of the field. To that end, we assume that a well-defined mechanism exists via which nodes are able to discard obsolete samples, 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. In our model, 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 data.

III. Payoff of Cooperation

Our goal in this section is to gauge the ability of a single node to answer queries within a given deadline. We present analytical results, which we validate by simulations. Our results show that the ability of a single node to answer time-constrained queries is limited. In particular, for a single node to successfully meet query deadlines, such deadlines must be quite loose – namely, in the order of $O(L^2)$, where L is the size of the field (*e.g.*, length of a 1-D field or area of a 2-D field). To meet tighter query deadlines, cooperation between multiple nodes is inevitable.

A. Single Node Performance

We present an analytical model that allows us to quantify asymptotically the time needed by a single node to reach *steady state*. We define steady state as the state of a node (*i.e.*, a walker) when it has sampled all locations in the field at least once, which translates into a 100% success ratio in answering queries that target random locations in the field. To be amenable to analysis, our model makes simplifying assumptions about the field, the mobility model of the nodes, and the size of the node caches. These assumptions are later relaxed in event-driven simulations (Sections VI and VIII), which confirm that our conclusions hold under more realistic assumptions.

In our analysis below, and for ease of presentation, we consider a one-dimensional (1-D) field, noting that our analysis could be extended in a fairly straightforward manner to higher dimensions, with *identical* conclusions.

Without loss of generality, we assume that the node[2] starts at location 0, and performs an m-step random walk in a transitionally invariant discrete field[3] of size L. A transitionally invariant 1-D field is a ring of length (perimeter) L. At each step of the random walk, the node samples the location at which it ends up. For the sake of simplicity, we

[1] This type of workload is the hardest to consider since the performance of any protocol can only benefit from *a priori* knowledge of the workload.

[2] We use the terms "node" and "walker" interchangeably.

[3] Assuming this type of a field spares us the need to handle halting and reflecting states in the field, which would complicate the analysis [28].

assume that nodes have infinite caches, which allows them to keep all collected samples locally. Clearly, the probability of a single node answering a time-constrained query is a function of the deadline of this query. Our analysis aims to derive asymptotically this probability.

A query is successfully answered if the node has a reading collected at a location that is at most ℓ distance units away from the location targeted by the query. For the sake of the analysis, we let $\ell = 0$. Later, in our simulation studies, we will relax this assumption and use $\ell > 0$.

Clearly, under the unlimited cache assumption, all queries could be answered as long as their deadlines are long enough. The question is how much time is "enough time". We argue that t_{ss}, the time needed for a node to reach steady state (*i.e.*, sample the entire field) as defined above, could be used as a gauge for deadlines of queries that could be answered by this node. In particular, if a query is received at time 0, then t_{ss} quantifies the time needed to answer the received query with a probability approaching 100%. If a query has a tighter deadline than t_{ss}, then, clearly, the probability of successfully answering the query by its deadline will be lower. Note that in this model, freshness of the answer is not a factor, since we assume that the query arrives at time 0, and once the requested sample is collected, it is delivered instantaneously as the answer.

The mobility model of the walker is an important factor that affects t_{ss}. Mobility models are defined in terms of the single-step distribution, $p(j)$. Assuming the node starts from location 0 in the field, $p(j)$ gives the probability that in one time unit, the node moves to another location that is j units of distance away. For a periodic unidimensional field (*i.e.*, a ring) of size L, we have $0 \leq j \leq L-1$. To illustrate this point, consider two nodes moving on a ring with two different mobility models. The first moves one step to the right every time unit, *i.e.*, $p(1) = 1, p(i) = 0, \forall i : 0 \leq i \leq L-1, i \neq 1$. While the other node moves either one step to the right or one step to the left with equal probability, *i.e.*, $p(1) = p(-1) = 0.5, p(i) = 0 \ \forall i : 0 \leq i \leq L-1, i \neq 1, i \neq -1$. Clearly, in a given period of time, the first node will cover more locations in the field than the second node.

We denote the number of uniquely visited locations after time t_x in a field of size L as $U_L(t_x)$. This allows us to define t_{ss} formally using the following equation.

$$t_{ss} \ = \ \{\min t_x | U_L(t_x) = L\} \tag{1}$$

Equation 1 states that t_{ss} is the time at which the node has visited each location in the field at least once.

Weiss gives the expected value of $U_L(t_x)$ in one dimension as (see Equation 4.191 in [28]):

$$E(U_L(t_x)) = L(1 - [1 + \frac{1}{L \ \phi_L}]^{-(t_x+2)}) \tag{2}$$

where ϕ_L is given by (see Equation 4.169 in [28]):

$$\phi_L = \frac{1}{L} \sum_{s=1}^{L-1} \frac{1}{1 - \hat{p}(\frac{2\pi s}{L})} \tag{3}$$

where $\hat{p}(\frac{2\pi s}{L})$ is the Fourier series of the single-step displacement function, $p(j)$, and is given by

$$\hat{p}(\frac{2\pi s}{L}) \equiv \sum_{j=0}^{L-1} p(j) \ \exp(\frac{2\pi i j s}{L}) \tag{4}$$

Keeping in mind that queries target random locations in the field, then dividing both sides of Equation 2 by L, allows us to calculate the probability $P(d)$ of a single node successfully answering queries as a function of the query deadline d.

$$P(d) \sim (1 - [1 + \frac{1}{L \ \phi_L}]^{-(t_x+2)}) \tag{5}$$

Notice that $1 - P(d)$ represents the probability of missing a deadline. In order to use Equation 5, we need to calculate ϕ_L, for which we need to calculate $\hat{p}(\frac{2\pi s}{L})$, which in turn needs $p(j)$ to be calculated.

The single-step displacement for the random walk, $p(j)$, characterizes the mobility model of the node. For the purposes of our analysis, we use a very simple process as the single-step displacement of the node – a drunkard walk [28] with no absorbing or reflecting states. In this walk, at each unit time, the node moves either one step to the left or one step to the right with the same probability. We call this walk, drunkard walk with pace = 1. Here the pace of the walker could be seen as modeling the maximum "speed" of the walker – the maximum absolute distance the walker is able to move in a single step (or unit time).[4]

To quantify the effect of the walker speed, we also consider a second mobility model under which the walker might move one, two, or three steps either to the left or to the right with the same probability. We call this model a drunkard walk with pace = 3. Under this model, the walker may travel at one of three speeds with equal (uniform) probability.

We also consider a third mobility model under which the various speeds of the walker are not selected uniformly. In particular, the node moves one step in either direction with probability = 1/4, two steps in either direction with probability = 1/8, or four steps in either direction with probability = 1/8. We call this model non-uniform drunkard walk with pace = 4.

Notice that the latter two mobility models have the same expected value for the absolute single-step displacement (namely 2), which is twice the expected value of the absolute displacement of the drunkard walk with pace = 1.

Figure 1(left) shows the probability of missing a deadline ($1 - P(d)$) as calculated using Equation 5 for the three mobility models, for $L = 100$. Figure 1(left) also shows the results obtained from a discrete-event simulation of a single mobile node using the three mobility models we described above. Every point is the average of 100 runs, with 95%-confidence intervals shown. These results match closely the values calculated analytically. It is clear that the higher the expected one-direction displacement of the node, the more

[4]For the symmetric walks we consider, $p(j) = p(-j)$, so the imaginary parts in Equation 4 readily cancel each other out.

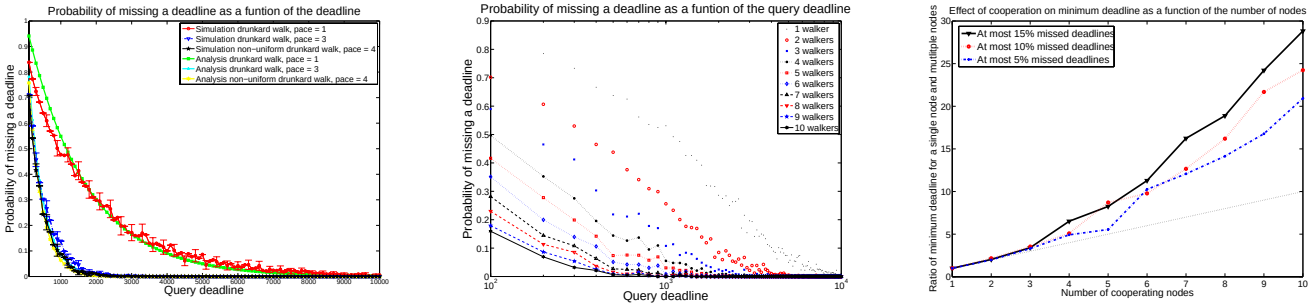

Fig. 1. Cooperation Premium: Probability of missing a deadline as a function of the query deadline for a single node (left) and for multiple nodes on a semi-log scale (middle), and the ratio between the minimum deadline required to achieve a given performance without and with node cooperation as a function of the number of nodes (right).

space it can cover in the same amount of time, which translates to a lower probability of missing a deadline. We also note that the performance of the drunkard walk with pace=3 and that of the non-uniform drunkard walk with pace=4 are quite close, suggesting that the first moment (*i.e.*, the mean) of the distribution of absolute displacement plays much more dominant role when compared to higher moments.

Both the analytical and simulation results shown in Figure 1(left) underscore the conclusion that a single node would be able to meet all deadlines, if such deadlines are proportional to $O(L^2)$. Clearly, this is impractical, especially for large fields. In the next section, we show that this limitation could be circumvented by allowing the mobile node to cooperate with other mobile nodes in the field.

B. Multiple Node Performance

In this section, we look into the interplay between spatial and temporal coverage of a field. In particular, *could one "buy time" by having more nodes roam the field and cooperate in responding to queries?* By "cooperation" we mean that the performance of the system is measured by the collective performance of all of its nodes. More specifically, if any node is presented with a query that it cannot answer based on the content of its cache, then that node could use other nodes in the system to answer such a query. If an answer for the query could be found in any node, then this query is counted as a success.[5]

We do so by repeating the experiments presented in the previous section with a different number of walkers. Assuming that a query is submitted at time $t = 0$, we count the number of uniquely visited locations by all walkers at time $d > 0$. The ratio between this number and the field size is $P(d)$, the probability of the system answering queries subject to a deadline d. Then, the probability of missing a deadline is calculated as $1-P(d)$. Figure 1(middle) shows the probability of missing a deadline. The parameters of this experiment match those of the previous experiment – a drunkard walk with pace=1 in a field of size $L = 100$. Initially, all nodes are placed uniformly at random over the field. Each point in Figure 1(middle) is an average of 50 simulation runs. As the number of nodes increase, the performance of the system improves (*i.e.*, it is able to meet stricter deadlines) and the system approaches steady state (*i.e.*, no missed deadlines) faster.

As shown above, having more nodes cooperate will allow the system to be more responsive to tighter deadlines. The question is whether the payoff from such cooperation goes beyond a simple tradeoff of space and time. One way to answer this question is to calculate the minimum deadline that results in a particular success rate for a single walker and the minimum deadline that results in the same success rate for i walkers. If the minimum deadline for a single walker is simply i-times that of i walkers, then we have a simple, linear tradeoff of space (number of walkers) and time (minimum deadline). Figure 1(right) shows the ratio of the minimum deadlines for a single walker and for i walkers when the success rate is 5%, 10%, and 15%. In all cases, we observe that the ratio is superlinear in i – *i.e.*, the premium from cooperation goes beyond a linear trade off of space for time.

It is important to note that the above results hold as long as the placement of nodes in the field is not correlated (or equivalently, that the mobility model of the nodes will not result in a skewed coverage of the field). In real settings, this may not be possible to guarantee, which in turn will impact (either positively or negatively) the benefits from cooperation.

IV. Cooperation Strategies

As discussed above, cooperation between multiple nodes is bound to improve the performance of the whole system. This cooperation, however, can take different forms. In this section, we propose two techniques to address the specifics of this cooperation. The two techniques are at the opposite ends of the scale with respect to using multi-hop communication.

A. Direct Placement and Retrieval (DPR)

DPR relies on multi-hop communication to both insert samples in the system and query them. Using hashing techniques, gathered samples are hashed to certain node(s) or location(s) in the field. Then, using either ad-hoc routing or geographic routing (depending on whether we hashed data objects to nodes or locations), data items can be forwarded to the hashed node or the node closest to the hashed position. This technique resembles previous work [23], [25], and depends on hashing and explicitly forwarding both samples and queries to their

[5]Here, and for the sake of quantifying the premium from cooperation, we abstract out the process via which such answer could be found in a real system. These considerations are the subject of later sections.

hashed nodes. We call such nodes the *hosting* nodes. This idea, in effect, partitions the set of data objects (in this case, samples gathered over the whole field), using a hashing function, and stores each partition in a separate node or group of hosting nodes. When a node gets a query, it hashes the query target to get the hosting node or group of nodes to such a query. Successful handling of queries is conditioned on successfully reaching one of the hosting nodes. This technique gives each node a very narrow but detailed view of the field; most samples kept by any node are concentrated in one small area in the field, which represents the *responsibility region* of this node. For this technique to be useable, query deadlines should be lower bounded by the communication delay over multiple hops. Clearly, if query deadlines are in the same ballpark as 1-hop delays, then the performance of DPR will be severely limited. Since the longest path in the field is a function of the communication range, node density, and field dimensions, determining the exact lower bound on the query deadlines depends on the parameters of the system.

B. Amorphous Placement and Retrieval (APR)

APR makes no use of explicit multi-hop communication, but relies on node mobility to diffuse data around. In this paper, we adopt a setting in which nodes keep samples that are gathered locally, and upon encountering another node, we use some shuffle exchange technique to diffuse the readings throughout the system. Nodes can assign some measure of value to each sample in their caches. One measure of importance or value of a sample is the minimum distance between this sample and all other samples in the cache. If this distance is large, this means that this sample is the only sample available locally for a large area, *i.e.*, this sample covers a wide area of the field, which means that it is a "valuable" sample. The method of selecting which set of samples to exchange with a neighbor when encountering them results in different flavors of this algorithm. One flavor – and the one we use in this paper – requires nodes to sort their samples based on some measure of value, and when encountering a neighbor, they exchange a fixed number of their most-valuable samples.[6] Upon receipt of a query, a node attempts to answer the query from its local cache. If an answer could not be found locally, only direct (one-hop) neighbors are queried for answers.

With APR, nodes have a broader view of the field (since shuffling aims at giving each node a "sample" of the entire field), which enables nodes to answer queries by querying their local cache or the cache of their direct (one-hop) neighbors – *i.e.*, without having to forward queries over multi-hop routes. By allowing nodes to leverage their mobility to *proactively* construct a broad view of the field, which is possible to query locally (or within a single hop), APR makes it possible for the system to be useful even when it is temporarily partitioned.[7]

[6]In [21], we have experimented with various policies for the selection of samples to be excahnged (*e.g.*, random, most/least valuable, ... *etc.*) in a non-real-time setting.

[7]Another advantage of APR (over DPR) relates to the inherent redundancy of data stored in the nodes, which translates into resilience in case of node failures or network partitions. The consideration of this is beyond the scope of this paper.

In DPR, each node has a unique region of the field for which it is responsible. In case of node failures, samples that are the responsibility of such failed nodes become inaccessible, causing the performance of the whole system to deteriorate (not to mention the impact of failures on the fidelity of the muti-hop communication which is required for placement and retrieval under DPR).

This problem is avoided in APR, where each node gets a global view of the field. Of course, DPR could be made more resilient to node failures by deliberately introducing redundancy; creating shared responsibility regions by more than one node, so failure of any single node wouldn't affect the operation of the whole system. This however will introduce complexities and overheads. In the remainder of this paper, we do not consider "failures", but we note that APR has an inherent advantage when it comes to failure tolerance.

The higher tolerance of APR to network partitions and node failures (when compared to DPR) comes at a cost. Namely, the storage capacity of all nodes is not utilized efficiently in APR (when compared to DPR). Indeed, as we will show later, APR is likely to underperform DPR for queries with higher precision requirements (*i.e.*, smaller ℓ).

V. Experimental Setup

Simulation Model: We use a custom-developed event-driven simulator. Our simulator allows us to plug-in various mobility models for the nodes as well as different routing and caching capabilities. It does not, however, model MAC layer details (*e.g.*, collisions, retransmissions). To allow for fair comparison, we used the same parameters for each protocol including the mobility model, the query model, and the model used to generate the reading samples.

In order to evaluate techniques that require multi-hop routing (namely, DPR), we implemented multi-hop routing based on Dijkstra's shortest path algorithm. Namely, when a node needs to send a packet to another node that is not within communication range, we use the shortest path routing algorithm to figure out the shortest path between the two nodes, if any. Since using Dijkstra's shortest path algorithm requires *instantaneous* global knowledge of the topology – knowledge that any realistic distributed mobile ad-hoc routing protocol would be lacking – results of any realistic implementation of DPR will be inferior to those shown here in terms of its ability to meet deadlines which translates into lower success ratio.[8]

Mobility Model: The mobility model we employ is the Random Waypoint mobility model sampling from the correct distribution to guarantee stationary speeds [15]. The minimum speed of the nodes is 1m/second, while the maximum speed is 20m/second, with pause time set to 0 to increase the dynamics of the simulation.

Baseline Parameters: Unless otherwise stated, results presented in this paper are for experiments with the following

[8]We note that for the purposes of this paper, this is prudent since our aim is to show under what conditions amorphous approaches (such as APR) outperform DPR, especially in sparse networks.

parameterization. We set the field size to be $L \times L$, where $L = 1400$ meters (slightly less than a square mile). We assume that a total of 100 nodes are roaming this field, each with a cache that holds up to 50 samples.[9] As we will show later, the cache size of the individual nodes ceases to affect performance once the cache size gets to be "large enough" to hold the working set of the spatiotemporal query workload.

Unless stated otherwise, we use a precision $\ell = 80$ meters – *i.e.*, a reading is suitable as an answer to a query if it is within 80 meters of the target of the query (5.7% of L).[10]

We assume that both field sampling and query arrivals are Poisson processes with rates of 2 seconds and 10 seconds, respectively. The default value of the communication range is 160m. The time-to-live for any sample is 200 seconds. This value, along with the sample insertion rate, ensures that caches will not be empty due to expired samples.

Performance Metrics: To evaluate the relative performance of the various protocols we consider, we use two metrics: the deadline miss ratio and the average precision of successfully answered queries. The *deadline miss ratio* is the ratio between the number of queries that could not be successfully handled by their deadlines and the total number of queries to the system during the simulation time. The *average precision of successfully answered queries* is the mean distance between the location of a returned query answer and the location specified as the query target. Notice that this value is upper bounded by the precision of the query ℓ.

Unless otherwise stated, for the figures in the remainder of this paper, each point represent the average of 20 simulations, with the 95% confidence intervals shown. The simulation time is 5000 seconds with measurements after a 150-second warm-up period.

VI. Baseline Performance of DPR and APR

Generally speaking, query deadlines may belong to three very different time scales: (1) they may be extremely tight, in the sense that they are in the same ballpark as the round-trip delay between neighbors within communication range R (*i.e.*, a delay consistent with a few hops), (2) they may be moderately tight, in the sense that they are in the same ballpark as the maximum round-trip delay between any two nodes in the field (*i.e.*, a delay consistent with enough hops to span the diameter of the field), or (3) they may be loose, in the sense that they are significantly longer than communication delays (*i.e.*, a delay that is orders of magnitude larger than 1-hop delays).

[9]We argue that 50 is a reasonable cache size to experiment with for a number of reasons. First, RAM size in today's hand-held devices is around 64KB to 128KB. This RAM has to fit a lot of OS modules and programs, hence what is left for applications' data is much less. Second, many sensor modalities require significant storage per sample. For example, in an imagery sampling application, if each image is 1.5KB, then to keep 50 samples we need 75KB of cache which is reasonable given today's standards. Lastly, memory chips of smaller sizes need less energy to refresh, which makes them more suitable for hand-held devices.

[10]Notice that the combination of cache size and precision we use in our baseline model imply that a single node is able to keep approximately 16% of all possible readings within the prescribed precision – for higher precision, the percentage is much smaller.

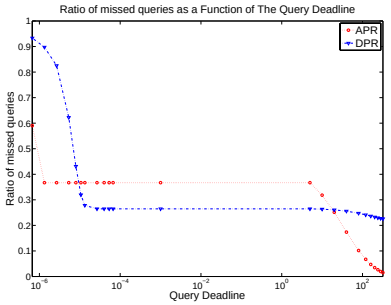

Fig. 2. Missed deadlines as a function of query deadlines.

We define *node reachability* as the set of nodes with which a node can successfully communicate within a given period of time – namely by the query deadline.[11]

Success of queries with extremely tight deadlines depends on the availability of an answer at nodes with very limited reachability. Notice that for all practical purposes, deadlines of such queries expire while nodes are still in the same locale, *i.e.*, the set of neighbors of any node will not change before such deadlines expire. On the other hand, success of queries with moderate or loose deadlines could leverage a wider set of nodes, including nodes from different locales, since nodes have enough time to change their locations in the field over the time scale of these queries.

Figure 2 shows the deadline miss ratio of DPR and APR as a function of query deadlines (under the baseline parameterization given above). The x-axis is plotted in log-scale to accommodate deadlines from a wide range of time scales. The left-most set of deadlines are at very short time scales that do not allow for reachability much beyond immediate neighbors. The set of deadlines in the middle are at time scales that extend the reachability of a node to all nodes with which a path could be established. The right-most set of deadlines are fairly lax, allowing for node mobility to play a role in affecting the reachability of a node.

For queries with very tight deadlines, the performance of APR improves between the first and the second points to the left, which corresponds to 1-hop and 2-hop delays, respectively. Deadlines of 1-hop delay allow nodes to only check their local caches, while deadlines of 2-hop delay allow nodes to query their direct neighbors as well resulting in better performance. Relaxing the tight deadlines beyond 2-hop delays does not immediately benefit APR since the content of the local cache and the contents of the neighbors' caches are not likely to change within this tight deadline. As for DPR, performance under tight deadlines improves linearly until it reaches a plateau. As mentioned above, increasing deadlines results in wider reachability, which translates into better performance for DPR.

For queries with moderate or loose deadlines, nodes get multiple chances to find answers to pending queries. Also, the time scale of the deadlines may allow nodes to change their

[11]Here, we ignore the delay associated with MAC layer issues like packet collisions and retransmissions.

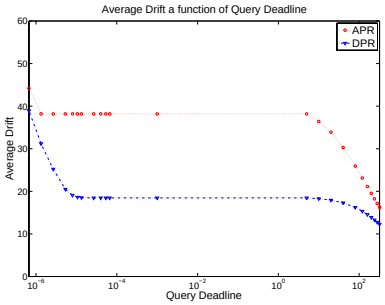

Fig. 3. Average precision of returned answers as a function of query deadlines

locale. Hence, if a node could not answer a query because it was isolated from the rest of the nodes in the field, it may become connected later and be able to answer such a query. Or, if an answer to a given query could not be found locally or at neighboring nodes, in the future, the set of neighboring nodes may change allowing multiple chances for answering the same query.

APR and DPR exhibit different behaviors when answering queries with moderate or loose deadlines. Loose deadlines do not result in significant improvement for DPR, but they do for APR. The reason is that for APR, longer deadlines allow nodes to consult different sets of neighbors, due to the mobility of the node (and its neighbors). As discussed above, this results in an improved performance of APR as the query deadline gets longer. This insight is consistent with results from our analysis, where we noticed that if queries have long enough deadlines, they will all be answered, eventually. On the other hand, for DPR, success of a given query hinges on successful communication between the inquiring node and the hosting node. Recall that this requires multi-hop communication, hence a path has to exist for this communication to succeed. If this communication is successful, then DPR achieves very high performance, otherwise the performance of DPR stabilizes at a low level. Increasing query deadlines is not likely to increase the likelihood that a path could be found between the inquiring node and the hosting node, since such connectivity is more or less a function of the density of the nodes over the field.

Figure 3 shows the corresponding average precision of the returned answer. For all values of deadlines from different timesclaes, DPR has the edge over APR. As mentioned earlier, DPR allows nodes to maintain a very detailed view of a specific region in the field. Thus, if an inquiring node is able to reach a hosting node, then the answer returned to the inquirer will be very precise. Notice that the precision of APR improves as query deadlines increase. The reason here is that as deadlines increase, an inquirer has a better chance of getting more precise answers to its queries by virtue of its ability to exchange samples with a larger set of direct neighbors over time, and hence improve its view of the field in anticipation of future queries.

To summarize, the results of this set of experiments show that for extremely tight deadlines in the ballpark of the communication delay between neighbors, APR outperforms DPR, but that for moderately tight deadlines in the ballpark of the communication delay over the diameter of the ad-hoc network, DPR works better. For loose deadlines that are orders of magnitude larger than communication delays, APR eventually outperforms DPR as node mobility results in wider field coverage.

For our envisioned applications, we believe that queries with extremely tight deadlines are not of great importance. Hence, in the remainder of this paper, we concentrate on queries subject to moderately-tight to loose deadlines – *i.e.*, restricting our consideration to timescales that are much larger than communication delays.

VII. A Hybrid Placement and Retrieval Protocol

In previous work of ours [21], we studied the main factors affecting the performance of both APR and DPR for non-real-time queries. We summarize our conclusions from that study below.

The main factor that affects APR's performance is the query precision ℓ. APR delivers very high performance for queries with relaxed precision requirements, while its performance suffers when the precision requirement becomes tight. The reason is that, under APR, each node gets a broad view of the whole field. This view has a much higher probability of satisfying queries, if the precision requirements for these queries were not very tight. When the precision requirement is tight, both local samples and samples at direct neighbors become less useful in satisfying such queries.

The main factor that affects the performance of DPR is the connectivity of the underlying network. Connectivity can be expressed in terms of the density of nodes in the field, and communication range of the nodes. When the underlying network is well connected, DPR delivers its best performance; otherwise, its performance degrades.

In addition to the above, our results from the previous section indicate that when query deadlines are moderate, DPR outperforms APR, while the roles are reversed for looser deadlines.

In a real setting, any of the system's baseline parameters (*e.g.*, query precision ℓ, communication range, node density, and query deadlines) may change dynamically. As a result, choosing either DPR or APR may backfire. Instead, what is needed is a placement and retrieval strategy that adapts its behavior towards either APR or DPR as appropriate in order to maximize performance.

With that goal in mind, we propose a Hybrid Placement and Retrieval (HPR) strategy. Locally, at each node, HPR independently adapts its behavior by following either APR or DPR for each type of data/query. We distinguish between data types based based on a dynamically computed threshold values. Data queried subject to deadlines that are below the computed threshold are handled using DPR, whereas those queried subject to deadlines that are above the computed threshold are handled using APR.

The computed threshold is adjusted using a gradient decent-like algorithm to minimize the ratio of missed dead-

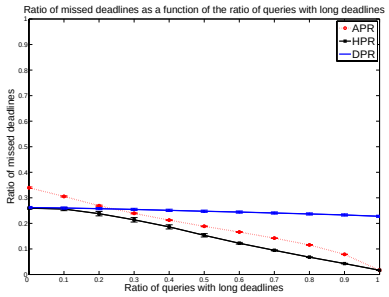

Fig. 4. Missed deadlines versus percentage of queries with long deadlines.

lines. Notice that since each of the baseline parameters in our system cause the system performance (for either DPR or APR) to monotonically improve or deteriorate, the optimal value reached by such an adaptation must be global (*i.e.*, the system would not be trapped in a local minimum).

The adaptation of the computed threshold allow us to react to a dynamically changing setting (*e.g.*, parameters affecting node reachability). In this paper, we do not consider the effects of such dynamics on the convergence of HPR, noting that the convergence of HPR is guaranteed as long as such dynamics occur at a time-scale that is long enough for our adaptive controller to converge to an appropriate threshold.

Because of the distributed nature of HPR, nodes may end-up with different deadline threshold values. Such control conflicts may be resolved in different ways, *e.g.* neighboring nodes may periodically reset their deadline thresholds to their average value.

VIII. Effect of Baseline Model Parameters

In this section, we evaluate how fhe performance of the three protocols (APR, DPR, and HPR) is affected by each one of the baseline model parameters. In particular, we focus on the effect of query precision, communication range, and cache size. We do so using a set of workloads that feature different mixes of tight and loose deadlines.

Effect of Deadline Mix in the Workload: We evaluate the proposed protocols using workloads in which queries represent a mix of moderately-tight and loose deadlines. In particular, we subject the system to two different types of data, whose query deadlines are one second and 320 seconds, respectively (seconds versus minutes timescales). Figure 4 shows the performance of the three protocols as a function of the percentage of queries with long (320-sec) deadlines in the workload.

Figure 4 underscores our earlier findings—DPR delivers better performance for queries with moderately tight deadlines, and APR excels when queries have looser deadlines. The performance of HPR could be seen as a linear combination of APR and DPR: it behaves like DPR for queries with tight deadlines and like APR for queries with loose deadlines, and as a linear combination of the two based on the threshold used to select either DPR or APR for a given data type.

In the remainder of this paper, we use three representative workloads to test our various protocols. These workloads feature different mixes of queries with tight and loose deadlines. In these mixes the ratio of queries with tight (1-sec) deadlines to queries with loose (320-sec) deadlines are 90:10, 50:50, and 10:90, respectively. We use this workload to evaluate the performance of the protocols as we change the parameters of the baseline model.

Impact of Query Precision: Query precision, ℓ, determines the range around a query target within which any sample is considered to be a valid answer. Figure 5 illustrates the performance of the three protocols (APR, DPR, and HPR) as a function of query precision, under the three representative workloads. In this experiment, the communication range is set to 160m and the cache size is set to 50. For high-precision queries (i.e., small values of ℓ), DPR is superior to APR. However, as query precision is relaxed, APR eventually outperforms DPR. HPR adapts its performance based on the observed deadline miss ratio to achieve the best performance—for high precision queries, HPR mimics DPR, whereas for lower precision queries it behaves like APR.

Impact of Communication Range: The communication range determines how well the network is connected. Figure 6 shows the performance of the three protocols as a function of the communication range. In this experiment, the query precision is set to 80m and the cache size is set to 50. Figure 6 show that DPR delivers its best performance in a very well connected network. When network connectivity deteriorates, so does the performance of DPR. APR outperforms DPR in networks with less connectivity. In well-connected networks, its performance is still good, but is inferior to that of DPR. In all communication ranges, HPR succeeds in following the correct protocol to minimize the percentage of missed deadlines.

Impact of Cache Size: Generally, the more cache available to nodes the better the performance should be. In this experiment we study the effect of the available cache on the different protocols. Figure 7 shows the performance of each protocol under the various workloads, using communication range of 160m and the query precision of 80m. Figure 7 shows that the performance of APR improves as we increase the available cache, whereas that of DPR is almost constant. The reason is that increasing the cache size, allows nodes to keep more samples locally, hence increasing the chance of answering more queries locally. Also, more queries can be answered using the neighbors' caches. For DPR, performance depends mainly on successful communication between the inquirer and the host node, which is not a function of the cache size, hence, its performance is almost constant. HPR has the minimum ratio of missed deadlines in all cases.

IX. Related Work

Data and Query Management in Sensor Networks: Data Centric Storage (DCS) [26] along with Geographic Hash Tables (GHT) [23] have been proposed as a good data dissemination technique for one-shot queries. In DCS, sensory data is hashed to a geographic location in the field. Using GPSR [13], this data is then forwarded to the node closest to the

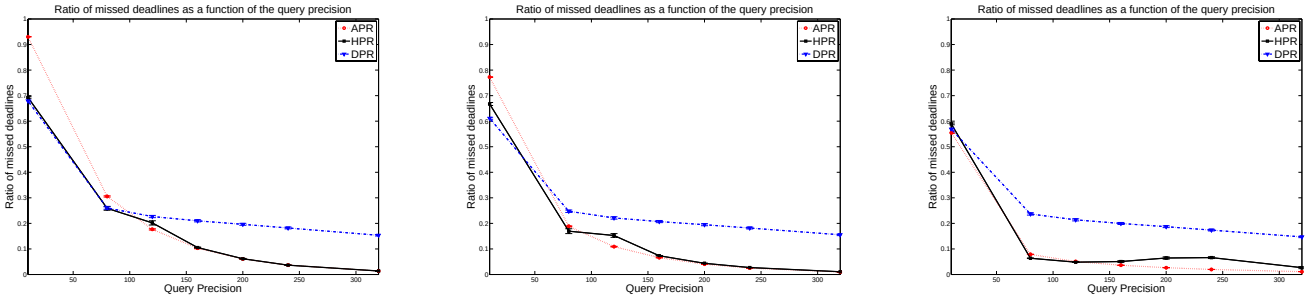

Fig. 5. Effect of query precision on probability of missing deadlines with 10% (left) 50%(middle), and 90%(right) queries with long deadlines.

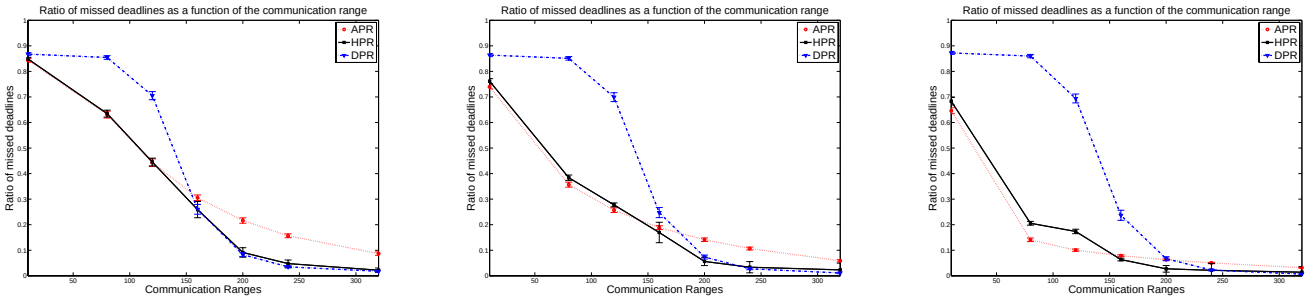

Fig. 6. Effect of communication range on probability of missing deadlines with 10% (left) 50%(middle), and 90%(right) queries with long deadlines.

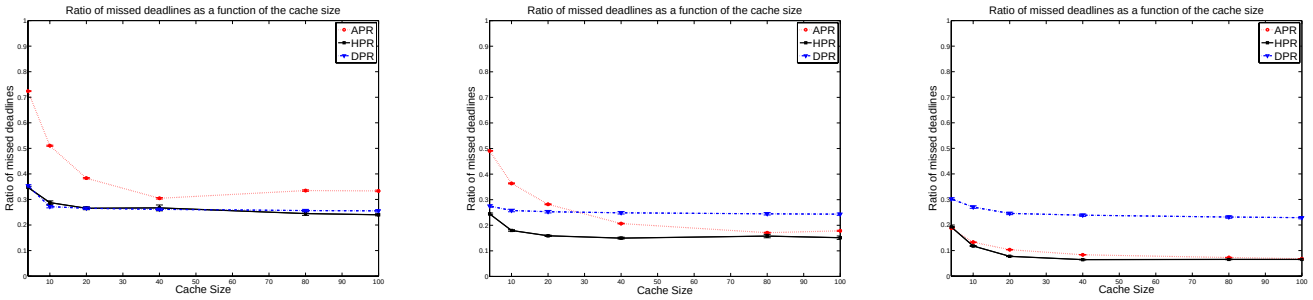

Fig. 7. Effect of cache size on probability of missing deadlines with 10% (left) 50%(middle), and 90%(right) queries with long deadlines.

target location for storage. Queries are similarly hashed and forwarded to the node where the answer is expected to reside. DCS is not primarily intended for systems with high mobility – since mobility changes the hashing of data items to nodes in the field, restoring the correct state of the system incurs large communication overhead. To alleviate this problem, Seada *et al.* [25] relax the hashing function from mapping data to a fixed *point* in the field, to mapping it to a fixed *region*. Data replication within the same region is used to minimize the communication overhead incurred when a node leaves the region.

The DPR technique we considered in this paper could be seen as an "ideal" version of DCS, and hence a representative of DCS approaches. We say ideal, because we assumed that routing of data to region for storage, and routing of queries and responses are done using global instantaneous shortest-path information, which is not possible in a real setting. Although it has been foreseen that the performance of DCS-like systems would depend on the connectivity of the underlying network, such sensitivity of performance has not been evaluated before. We evaluate this sensitivity and propose alternative protocols (APR and HPR) that outperforms DCS in sparse mobile networks.

Data Centric Routing (DCR) techniques have been proposed for *static* sensor networks [10], [5], [19]. In these techniques, collected samples and detected events are locally stored. Sinks (originators of queries) have to declare their queries to the network to collect results. Flooding has been the primary solution to route queries over the network, since it is not known which node/locale of the network might host an answer to any query. In such techniques, flooding has been regarded as a waste of sensors' energy, a crucial resource in sensor networks. Multiple research efforts focused on avoiding such flooding to save energy and increase the network lifetime. [29], [14] are two such examples. Assuming only sink mobility, Ye *et al.* [29] propose the TTDD system, in which, the network constructs a grid overlay over the sensor field. This grid is used for storing sink queries, gathering events from source nodes, and delivering the results back to the mobile sink. Under similar assumptions, Kim *et al.* [14] propose SEAD, an asynchronous dissemination protocol. In SEAD, the network builds a dissemination tree rooted at the

sensor closest to the current sink location. The tree is used to send events back to the sink. The system has to strike a balance between the cost of rebuilding the tree as the sink changes its location vs. communicating results from the existing tree to the sink over multiple hops.

The APR technique we propose in this paper follows the same concept of DCR in locally storing samples in nodes that gathered them. However, our design does not have to resort to network-wide flooding to find query answers, rather we employ only 1-hop flooding which proves to deliver very good performance due to the proactive nature of APR in forming a broad view of the field. Moreover, unlike previous work, we assume that the whole network is mobile which poses a harder challenge to overcome.

Handling Queries with Spatio-temporal Constraints: Systems that are capable of handling queries with both spatial and temporal constraints have been proposed [6], [16]. However, in these works the definition of spatial constraints is different from ours. In [6], nodes are *static*, and queries have a spatial window. All nodes in such a window are required to participate in answering the query. On the other hand, in [16], Lu *et al.* associate the spatial window of a query with the current sink location, *i.e.*, they assume that sinks are only interested in their current locale. In our model, nodes are satisfied with a sample that is at most ℓ distance units away from a specific point in the field. Thus, we can support applications where the only sensor node hosting a requested sample (*e.g.*, a particular event detected in a certain area) may be mobile. Furthermore, a query can target any location in the field, including locations that the node originating the query has not visited.

The protocols we considered in this paper require nodes to know their locale to be able to attach location information to samples they collect. Many localization protocols have been developed for sensor networks [20], [11], [8], and any of these could be used to supply the needed location information. Of course, our protocol could also make use of a global positioning system (GPS), especially since the personal communication devices, which we envision as hosting the sensors, are likely to have such GPS capabilities.

X. Summary and Conclusion

In this paper, we investigated the potential use of local caches in mobile, ad-hoc sensor networks for the collective storage and querying of sensory data collected by mobile nodes. Using a simple analytical model, we made the case for node cooperation and showed that its benefits go beyond a simple space-time trade off. To reap the benefits from cooperation, we argued that a Data Centric Storage (DCS) approach that uses direct placement and retrieval (DPR) is viable, but only when the underlying network is well connected. Alternatively, we proposed an amorphous data placement and retrieval approach (APR), in which sensory samples are cached locally and *shuffling* is used to diffuse that data throughout the network. We leveraged insights gained from comparing the performance of APR and DPR to propose Hybrid Placement and Retrieval (HPR). Using extensive simulations, we showed that HPR is able to adapt its performance, delivering the least rate of missed deadlines under most conditions.

Our current research is focusing on cache management techniques that allow nodes to leverage their knowledge of underlying mobility models (*e.g.*, locality characteristics), as well as the spatiotemporal characteristics of the underlying phenomena being sensed (*e.g.*, using summaries for a more effective exchange of readings). Also, we are investigating the implementation of the techniques presented in this paper (and variants thereof) in real personal communication devices to answer queries related to field conditions (*e.g.*, "what is the network coverage or signal strength in location x" or "how many different people are observed in location y").

References

[1] Lifeguard: Wearable Wireless Physiological Monitor. http://lifeguard.stanford.edu/.

[2] Man–Portable Networked Sensor System. http://www.spawar.navy.mil/depts/d30/d37/d371/mpnss/mpnss.html.

[3] Portable Sensors Extend Warriors Reach. http://www.afcea.org/signal/articles/templates/SIGNAL_Article_Template.asp?articleid=1114&zoneid=181.

[4] M. Ali and Z. A. Uzmi. CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks. In *2nd Annual Conference on Communication Networks and Services Research (CNSR'04)*, pages 165–174, Fredericton, N.B, Canada, May 2004.

[5] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *MDM '01: Proceedings of the Second International Conference on Mobile Data Management*, pages 3–14, London, UK, 2001. Springer-Verlag.

[6] Alexandru Coman, Mario A. Nascimento, and Jörg Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 104–110, New York, NY, USA, 2004. ACM Press.

[7] R. Cristescu, B. Beferull-Lozano, M. Vetterli, D. Ganesan, and J. Acimovic. On the interaction of data representation and routing in sensor networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Pennsylvania, USA, 2005.

[8] L. Doherty, K. S. J. pister, and L. E. Ghaoui. "convex position estimation in wireless sensor networks". In *IEEE INFOCOM*, volume 3, pages 1655–1663, March 2001.

[9] Deepak Ganesan, Deborah Estrin, and John Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev.*, 33(1):143–148, 2003.

[10] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom: Mobile Computing and Networking*, pages 56–67, 2000.

[11] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In *IEEE INFOCOM*, March 2004.

[12] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: the dynamic source routing protocol for multihop wireless ad hoc networks. *Ad hoc networking*, pages 139–172, 2001.

[13] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[14] Hyung Seok Kim, Tarek F. Abdelzaher, and Wook Hyun Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 193–204, New York, NY, USA, 2003. ACM Press.

[15] G. Lin, G. Noubir, and R. Rajamaran. Mobility models for ad-hoc network simulation. In *INFOCOM*, 2004.

[16] Chenyang Lu, Guoliang Xing, Octav Chipara, Chien-Liang Fok, and Sangeeta Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 381–390, Washington, DC, USA, 2005. IEEE Computer Society.

[17] 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.

[18] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[19] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[20] David Moore, John Leonard, Daniela Rus, and Seth Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM Press.

[21] Hany Morcos, Azer Bestavros, and Ibrahim Matta. Amorphous placement and retrieval of sensory data in sparse mobile ad-hoc networks. Technical Report BUCS-TR-2006-005, Computer Science Department, Boston University, 111 Cummington Street, Boston, MA 02135, April 2006.

[22] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. 2003.

[23] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. Ght: a geographic hash table for data-centric storage. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 78–87, New York, NY, USA, 2002. ACM Press.

[24] Ratnasamy S., Francis P., Handley M., Karp R., and Shenker S. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 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] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.

[27] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[28] G. H. Weiss. *Aspects and Applications of the Random Walk.* Elsevier Science B.V., North-Holland, 1994.

[29] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, New York, NY, USA, 2002. ACM Press.