Predictive QoS Routing to Mobile Sinks in Wireless Sensor Networks

Branislav Kusy, HyungJune Lee, Martin Wicke, Nikola Milosavljevic, and Leonidas Guibas
Stanford University, Stanford, CA, USA
{kusy,abbado,wicke,nikolam}@stanford.edu, guibas@cs.stanford.edu

ABSTRACT

We present an algorithm for data delivery to mobile sinks in wireless sensor networks. Our algorithm is based on information potentials, which we extend to account for mobility. We show that for local movement along edges in the communication graph, the information potentials can be adapted using a simple iterative distributed computation. However, for non-local movement, the potential field might change significantly, and iterative computation leads to packet loss and packet delivery delays. We address this problem by introducing the *mobility graph*, which encodes knowledge about likely mobility patterns within the network. The mobility graph can be extracted from training data and is used to predict future relay nodes for the mobile node. Using the mobility graph, we can precompute and efficiently store additional routing states in the network. This enables the algorithm to maintain uninterrupted data streams. We analyze the benefits of computing and maintaining a mobility graph, and show that the information contained therein can be used to improve routing reliability in experiments involving mobile sinks.

Categories and Subject Descriptors: C.2.4 [Computer -Communications Networks]: Distributed Systems

Keywords: Sensor Networks, Mobile Routing

General Terms: Algorithms, Prediction, Routing

1. INTRODUCTION

While routing in static settings is well-explored, routing in the presence of significant mobility must still be considered an open problem. This is particularly true for routing in wireless sensor networks, where high link volatility, scarcity of computational resources, as well as energy constraints further complicate the problem.

There have been significant advances in dealing with the above-mentioned problems in static sensor networks, and compact and reliable routing schemes have been developed [8, 28, 34]. These approaches, however, cannot compensate for the drastic connectivity changes that result from moving network nodes. Nevertheless, data delivery to a mobile node is an important problem in practice. For example, janitors can improve their efficiency by using live data in a building equipped with monitoring sensors, vineyard workers can access mildew, humidity, or chemical soil analyzing sensors during their daily check-ups in the field, or biologists can do exploratory research in the field with the help of habitat monitoring sensors.

In this paper, we address the problem of data delivery to a mobile node in a wireless sensor network. For this purpose, the sensor network is considered semi-static, and while individual links may experience high volatility, its long-term connectivity changes only gradually. The mobile nodes roaming through the network act as data sinks only, and are not used to forward information. Finding a route from anywhere within the network to a mobile node is accomplished by finding a route to a relay node in the network which is within radio distance of the mobile node.

Thus, the problem can be decomposed into two parts: finding a good relay node, and finding a route to the relay node. Network connectivity of a mobile user changes rapidly at higher speeds, and therefore setting up the route must be fast to avoid high latency or packet loss. Further, depending on the network topology, the route can change significantly even if the relay node moves only locally. Packets already en route will be lost unless additional mechanisms ensure delivery.

The routing algorithm that we present is based on a routing tree which is implicitly defined as the gradient of *information potentials* [26]. Our information potentials can be computed in a lightweight, distributed fashion. Their convergence, however, is slow and they have been thought to be impractical for moving sinks. We analyze the routing trees induced by information potentials and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IPSN'09, April 13-16, 2009, San Francisco, California, USA.
Copyright 2009 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

show that, if the movement of the sink is local, the information potential converges after very few iterations and defines a valid routing tree. If the movement of the sink is non-local, however, the potential field has to change significantly, requiring a large number of iterations until convergence. The problem of adaptation of routing data structures in situations when their local update is insufficient is interesting in a broader aspect. We are not aware of any previous work that analyzes when such situations occur and what steps can be taken to improve performance of routing protocols.

We introduce the *mobility graph*, a structure that encodes the movement patterns of mobile nodes. By analyzing its differences with the connectivity graph, we can determine which relay node transitions can be handled by iterative computation of the information potential and which transitions would require too many iterations to converge without packet loss. For such non-local transitions, we utilize precomputed information potentials. To further improve routing latency and reliability, we use the mobility graph to predict future relay nodes. Information potentials for these future relay nodes are then computed while the old route is still valid, enabling an instantaneous switch to the new relay node.

We evaluate the routing algorithm with and without prediction, and show that prediction significantly improves reliability. We present a detailed analysis of the effect of mobility on information potentials.

In summary, our contributions are the following:

- We introduce the *mobility graph*, a structure that encodes knowledge about the possible movements of agents roaming the area covered by a sensor network. We show how to compute a mobility graph from measured data, and demonstrate its utility for routing and prediction.
- We propose a method for mobility prediction that computes future relay nodes based on knowledge of the mobility graph, as well as RSSI traces that represent the edges of the graph. Experiments show that the best relay node for a mobile user can be predicted with up to 90% reliability seconds before the transition to that node happens.
- We extend routing based on harmonic information potentials [26] to support mobile nodes. We show that the routing trees induced by the information potentials quickly stabilize (after 3.8 iterations on average) for local movement of the data sink, making the approach practical for mobile nodes.
- We use the above to implement the predictive routing scheme optimized for data delivery in sensor networks. The prediction is used to compute new routing potentials before the old route becomes useless. We show in simulations that prediction increases the reliability of data delivery by up to 50% for mobile nodes at walking speeds. The algorithm is robust to prediction errors.

The remainder of this paper is structured as follows: After we discuss related work in Section 2, we address the problem and give an overview of our method in Section 3. Section 4 discusses the mobility model and introduces the *mobility graph*, before we describe our approach for prediction of future relay nodes in Section 5. We describe our routing scheme in Section 6. Finally, Section 7 presents an experimental evaluation of the approach and discusses the results.

2. RELATED WORK

Our work has two main components: mobility prediction and routing to mobile sinks. In the following, we discuss the most relevant related work in those areas.

Mobility Prediction.

We use prediction to find future relay nodes based on past observations. Similar problems have been approached using Markov-type models. Liu and Maguire [27] predict user movements in cellular networks. Paths are matched to a database using similarity measures taking into account positions, speeds, and recurrence frequencies. While their model is simple, its predictive power stems from its heavy customization to individual user profiles and large movement databases. Similar models have also been used for prediction in IP networks [13]. In simulations using the Dartmouth College wireless network usage data [20], the method improves packet reception ratios as well as latency. The experiment [32] collected Wi-Fi traces of mobile users for a period of 2 years. The authors have shown that Markov models can predict the next wireless access point with around 70% accuracy for a median user. Lempel-Ziv text compression algorithms were also successfully applied to local prediction problem by treating it as a strictly domain-independent problem in [4].

Markov-style predictors are particularly well suited for applications involving large networks in which users move in a repeatable, non-random way [22]. However, these algorithms use only a tiny fraction of the available data, ignoring signal strength measurements and performing the prediction based on current and past states alone. In smaller, less structured environments, using all available data is essential. Research in the field of robotics has developed localization and tracking algorithms based on signal strength measurements [29, 37].

An interesting class of algorithms uses clustering and path matching to determine likely future trajectories (e. g. [3, 14, 35]). These algorithms define routes through a collection of trips which are similar to each other. Similar to our approach, the prediction algorithm calculates similarity of the current trip to the past trips and returns the closest match.

Routing to Mobile Sinks.

In recent years, several routing protocols for networks of mobile nodes (MANETs) have been developed [6, 7, 30, 31]. These provide point to point routing capabilities in networks in which all nodes are considered mobile. While this requires only minimal assumptions on the mobility of nodes or traffic patterns, routes have to be recomputed frequently, an inefficient solution if parts of the network are static.

On the other end of the spectrum, data delivery protocols for sensor networks [5, 33, 38] assume (mostly) static networks that may be subject to link volatility. A lot of effort is spent on optimizing link and route quality estimates which makes these algorithms inefficient in adapting to mobile users.

Researchers have sought algorithms for special network setups involving mobile nodes: If the mobility of some nodes can be controlled, nodes can be moved to optimize energy efficiency of the network [12, 17, 36]. If the mobility can only be predicted, but not controlled, mobility can be used to transport data [18, 19].

In this work, we will treat a commonly seen case: a mobile sink communicates with the sensor network. While the sensor network nodes are static, the sink is not. For this problem, we need a data delivery supporting a moving root. Our routing scheme is based on information potentials [26]. While we assume no control over the movement of the mobile sink, we use (imperfect) prediction to compute fresh routes before old routes become useless.

The small state and weak synchronization requirements for information potentials are in sharp contrast to most other methods for data delivery to mobile users [1, 2, 10, 11, 24], which use global data structures.

3. OVERVIEW

We treat the problem of data delivery to a mobile sink in wireless sensor networks. Sensor nodes are thought to be static (i. e. not mobile), but otherwise subject to constraints typical for a wireless sensor network. In our approach, one sensor node, called the *relay node*, is designated as a proxy to the mobile node, through which the data is forwarded to the user. Thus the relay node becomes the data sink for all traffic destined to the mobile node. Data can become available at any network node, and at arbitrary times, so that all nodes must be able to send data to the relay node at all times.

Routes to the mobile user are defined implicitly via the gradient of information potentials [26]. The potentials are iteratively calculated in the network in a distributed fashion. Whenever the relay node changes, the information potentials have to be adapted.

We define the *mobility graph*, a data structure that allows for seamless transition to a new relay node, resulting in significant improvements in routing reliability. We describe how the mobility graph can be extracted from radio signal strength (RSSI) traces of users in the environment. By associating RSSI traces with edges of the mobility graph, we can predict future relay nodes and time to the transition. We use *dynamic time warping* (DTW) to match the current RSSI trace of the mobile node to the traces recorded in the past.

To minimize the impact of mobility, our routing algorithm updates information potentials for both the current and the predicted relay node, guaranteeing that the new information potential is ready once it is needed. We show that if the current and predicted relay nodes are close in network topology, the new information potentials require only very few iterations to define a valid routing tree. Thus in this typical case, the relay node prediction provides relatively small improvements of the routing algorithm.

Prediction is extremely beneficial when users move along edges in the mobility graph that correspond to multi-hop paths in the network. In these cases, iterative updates of network potentials does not yield good routes quickly enough. We therefore store precomputed information potentials at strategic points in the mobility graph. Thus, the routing algorithm can update its potential field to the precomputed field instantaneously, providing us with valid, near optimal routes to the relay node at all times.

4. MOBILITY PATTERNS

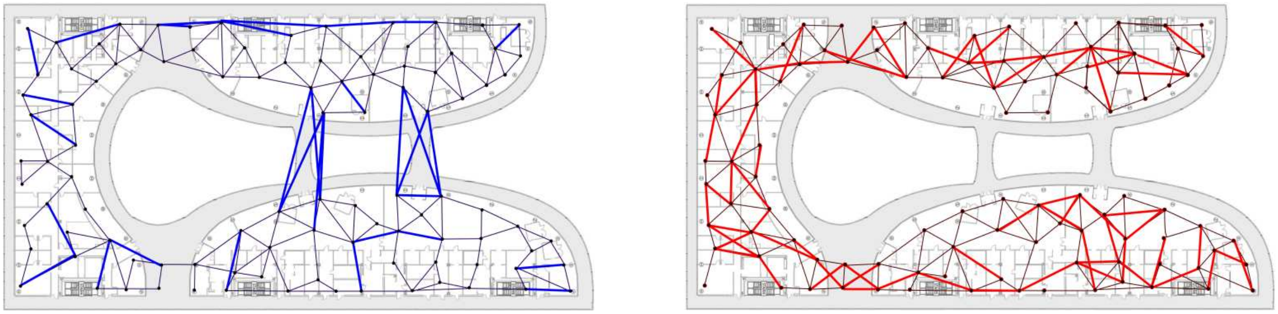
Assuming that we can observe users as they go about their everyday activities over a certain period of time, it is possible to extract mobility patterns of the users in the observed environment. In this section, we define the *mobility graph*, a data structure that encodes these mobility patterns.

More specifically, we assume that users carry a mobile computing device that actively communicates with surrounding sensor nodes. This allows us to observe trajectories of the users indirectly via RSSI traces[1] of their communication links. Formally, we assume that N cooperating infrastructure nodes measure signal strength $r_i(t), i = 1 \dots N$ for each packet from a user U received at time t. A location of the user at time t corresponds to an observation vector

$$R(t) = (r_1(t), ..., r_N(t)).$$

---

[1] Or in general, traces of any other link quality estimator.



(a) Mobility Graph

(b) Network Connectivity Graph

Figure 1: The differences between mobility and connectivity graph. (a) Note the additional (blue) edges in the mobility graph in regions where the network provides no coverage. (b) The connectivity graph is significantly more dense (red edges) in areas where movement is constrained by walls.

At each point in time, we define the *best connected node*

$$B(t) = \underset{i=1\ldots N}{\operatorname{argmax}}\ r_i(t)$$

as the node measuring the highest signal strength at a given point in time.

In the discrete setting, the *trajectory* of a user corresponds to an observation sequence

$$R(t_1 : t_k) = R(t_1)R(t_2)\ldots R(t_k).$$

Given such an observation sequence, we can define the sequence of best connected nodes

$$B(t_1 : t_k) = B(t_1)B(t_2)\ldots B(t_k).$$

Note that we do not assume the ability to measure locations of the users directly, nor do we assume any relation between the location of the user and the measured signal strengths. However, we do assume that if the user follows the same trajectory at different times, the corresponding observation sequences will be similar (after resampling). In essence, we assume that the environment does not change drastically over time and we aim to optimize routing protocols for the case of frequently repeated mobility patterns. Even though we require the training phase in our current approach, data aging techniques can be used to relax this assumption.

4.1 Mobility Graph

The mobility graph is a high level data structure that encodes mobility patterns learned from observation sequences. Formally, the mobility graph is defined on a set of N vertices, corresponding to the infrastructure nodes. Two vertices v_m and v_n are connected by a (directed) edge if there exists an observation sequence such that at some point, the best connected node switched from m to n:

$$\exists i : \{B(t_i) = m\} \wedge \{B(t_{i+1}) = n\}.$$

Intuitively, an edge in the mobility graph is inserted whenever the user moves from node v_m to v_n. This edge assignment essentially cuts the observation sequences into short segments, each segment corresponding to the transition between two nodes. For a trajectory R_i, each edge $e_{n\to m}$ connecting vertices v_n and v_m is associated with the segment $R_i^{n\to m} = R_i(t_{j_n}, t_{j_m})$ for which the best connected node is n:

$$\begin{aligned} B_i(t_{j_n-1}) &\neq n \quad \wedge \\ \forall t_{j_n} \leq t \leq t_{j_m} : B_i(t) &= n \quad \wedge \\ B_i(t_{j_m+1}) &= m. \end{aligned}$$

This defines a set of segments $\mathcal{S}_e$ for each graph edge e.

Although other ways of associating the data with the graph exist, we found this to be most useful for routing and prediction as described below.

Mobility Graph vs. Network Connectivity Graph.

Even though the mobility and network connectivity graphs are defined on the same set of vertices, they can have substantially different edge sets (see Fig. 1). On one hand, the mobility graph can contain edges that are not present in the connectivity graph, if a user moves through areas without network coverage. On the other hand, the connectivity graph can contain edges that are not present in the mobility graph, since radio signals can bypass obstacles, or travel through walls.

Using the Mobility Graph.

We use the mobility graph in two ways. The mobility graph constrains future location of a user in the network. We show in Section 5 that if we store typical signal strength observation sequences at edges of the mobility graph, both future relay nodes for a mobile user as well as the time to transition to that node can be predicted with high accuracy.

Edges in the mobility graph that do not exist in the connectivity graph are the basis for the predictive routing algorithm described in Section 6. We use the differences between the two graphs to identify regions in

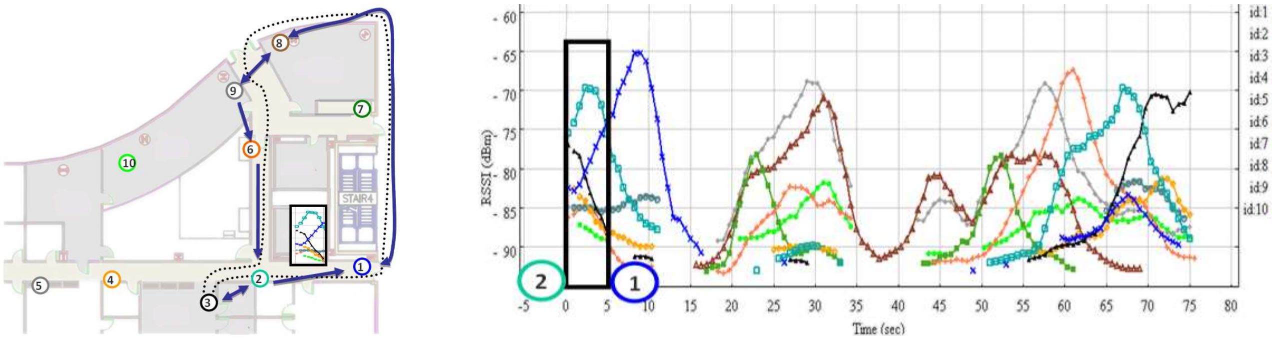


Figure 2: Extracting the mobility graph from an observation sequence. Left: The layout of our office with 10 infrastructure nodes. The trajectory is shown by a dotted line, arrows show the extracted mobility graph. Right: RSSI data recorded during the experiment. An edge between nodes 2 and 1 is highlighted. This corresponds to the data segment in which node 2 is the best connected node.

the network for which routing information needs to be precomputed.

We believe that the mobility graph is a valuable data structure, independent of the uses laid out in this paper. For example, the differences between the mobility and connectivity graph can guide network administrators in deploying additional nodes or redeploying existing nodes to improve the quality of the network coverage, or to understand data traffic patterns.

4.2 Mobility Graph Extraction

The mobility graph is extracted from a set of observation sequences $\mathcal{R} = \{R_i(t_{i1} : t_{i2})\}$ that correspond to users moving in the environment. We assume that the sequences are preprocessed and the set $\mathcal{R}$ contains only continuous, densely sampled observation sequences. An example of one such trajectory is shown in Fig. 2.

Since the vertex set of the mobility graph is defined by the set of infrastructure nodes, we only have to decide which edges should be present in the mobility graph. We determine the edges solely from the observation sequences. Given an observation sequence $R_i(t_1 : t_k)$, we add an edge in the mobility graph whenever the corresponding best connected node $B_i(t)$ changes.

In practice, this algorithm might construct a large number of edges in the mobility graph due to the noise in the link quality measurements. Mobility of the users exacerbates the effects of reflections and signal fading in structured buildings or other urban environments. We have implemented a number of filters that prevent constructing unnecessary edges.

The observation sequence is low-pass filtered and the nodes in the best neighbor sequence are retained only if they provide a high quality link for at least two seconds. The obtained mobility graph is further pruned by removing its infrequently observed edges. This final filtering step can be adapted to provide a simple yet efficient data aging mechanism: as new measurements are taken, rarely visited edges in the mobility graph are deleted, enabling the mobility graph to adapt to gradual changes in the environment.

The filtering criteria are motivated by a cost analysis of the routing algorithm. The cost of briefly losing a connection to a node (a link failed because we did not choose the "best" node with the highest signal strength) is much lower than the cost of setting up a new connection (if we switch nodes for less than two seconds). Other applications may dictate other criteria, resulting in a slightly different mobility graph.

5. MOBILITY PREDICTION

As we will show in Section 6, prediction of user mobility can significantly improve routing performance. We use the mobility graph as defined above to predict future relay nodes. We use pattern matching to determine the current position in the graph.

With each edge e of the mobility graph, we associate a set $\mathcal{S}_e$ of observation sequence segments that are representative for this edge. This set is determined in the training phase during which the mobility graph is extracted. Set $\mathcal{S}_e$ contains all segments that witnessed the mobility edge e, normalized to the same transmission power.[2] While the network is deployed, new data can be added to the graph, and data aging techniques can be applied to adapt the graph to gradual changes in the environment.

The prediction problem can then be stated as follows: At each time t, we know the current position of the mobile user in the mobility graph, given by the current relay node v_t. Given the current RSSI measurements $R(t_0 : t)$ since the last change of relay node, what is the next relay node and when will the transition occur?

Using the mobility graph extracted in the training phase, we can restrict the search by only considering

---

[2] We simply calculate the mean segment-wise RSSI value for each segment and scale the segments to have the same mean. This way, the mobility prediction algorithm works even if the transmitted signal had variable signal strength.

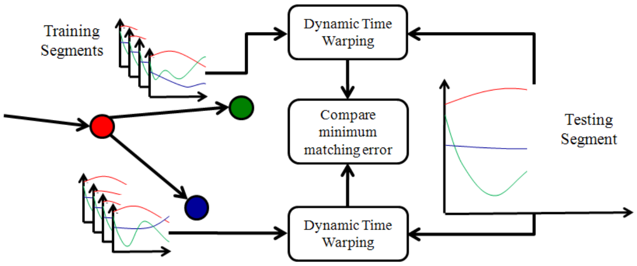


Figure 3: Dynamic time warping determines the best-matching outgoing edge and predicts the next relay node.

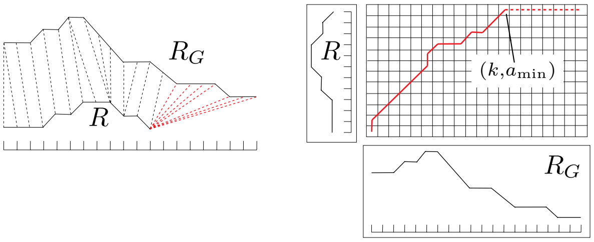


Figure 4: Dynamic time warping: Left: The best match between two sequences R and R_G defines pairwise matches between individual samples. If one sequence is significantly shorter, its last sample is matched to the remainder of the longer sequence (red correspondences). Right: The warp distance D_{DTW} in matrix form. The warp path W is shown in red. In case of a partial match, W reaches the edge of the cost matrix.

the set of outgoing edges $\mathcal{E}_{\rightarrow}(v_t)$ from the current graph vertex v_t. In order to determine the correct edge, we match the current RSSI trace to the stored RSSI segments of all edges in $\mathcal{E}_{\rightarrow}(v_t)$. The edge $e_{\min}$ associated with the best-matching segment is chosen, and its end-vertex is the predicted next relay node.

$$e_{\min} = \operatorname*{argmin}_{e \in \mathcal{E}_{\rightarrow}(v_t)} \min_{R_G \in \mathcal{S}_e} D_{\mathrm{DTW}}(R, R_G)$$

The distance function D_{DTW} is discussed below. Fig. 3 illustrates the matching algorithm.

5.1 Matching Segments Using DTW

We use dynamic time warping (DTW) [23] to find a reliable match in the presence of significant differences in speed. Since mobile users move through a dynamic environment, and each mobile user has different characteristics (the most obvious is different natural walking speeds), the measured RSSI sequence is warped non-linearly in the time domain. The DTW algorithm provides a similarity measure between two non-linearly warped sequences as well as the optimal pairwise match within the sequences. While the plain DTW algorithm assumes that both sequences are complete, in our case the current RSSI trace should be matched to the first part of the stored data. We therefore use a variant of DTW supporting partial matching. An additional output of the matching algorithm is the estimated time to transition.

In the following discussion, we will denote the current RSSI trace as $R(t_1 : t_k)$ and a trace stored in the mobility graph as $R_G(t'_1 : t'_l)$.

To compute a matching, we first calculate the matrix of element distances D by comparing each RSSI sample from R to each sample from R_G:

$$D_{ij} = d(R(t_i), R_G(t'_j)), \tag{1}$$

where $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, l\}$, and $d(\cdot, \cdot)$ is a distance function operating on vectors of RSSI samples; we simply use the $\mathcal{L}^2$ distance.

The warp path distance matching the first i samples of R to the first j samples of R_G can then defined recursively by

$$\begin{aligned} D_{\mathrm{DTW}}(i,j) = D_{ij} + \min \big[& D_{\mathrm{DTW}}(i-1,j) + \alpha, \\ & D_{\mathrm{DTW}}(i-1,j-1), \\ & D_{\mathrm{DTW}}(i,j-1) + \beta \big]. \end{aligned}$$

The penalties α and β are applied when a sample is skipped in the stored or current data, respectively. The traditional warp distance assuming a complete match is now $D_{\mathrm{DTW}}(k,l)$. The path taken by the recursion defines matches between individual samples in the sequences. We will write it as a sequence of index pairs $W = [(1,1), \ldots, (k,l)]$. Fig. 4 illustrates the matching.

By default, DTW matches the complete sequence and computes errors accordingly. This behavior favors sequences of equal length. Since we are interested in partial matches, we consider only the error incurred until R is fully matched: we find the first sample $a_{\min}$ of R_G that matches the end of R, $a_{\min} = \min\{a \,|\, (k,a) \in W\}$ (see Fig. 4 for an illustration). The final DTW distance for the partial match is then $D_{\mathrm{DTW}}(k, a_{\min})$.

In our experiments, we have obtained the best results penalizing stretching of longer sequences, and compression of shorter sequence. Thus, if $t_k - t_1 \leq t'_l - t'_1$, we use $\alpha = 50$ and $\beta = 0$, while otherwise we use $\alpha = 0$ and $\beta = 50$. The results, however, are not very sensitive to the choice of these parameters.

Expected Time to Transition.

In order to synchronously change the routing behavior throughout the network, it is useful to have advance warning when the relay node changes. Using the partial DTW matching outlined above, we can easily compute an estimate. As before, we are given the current RSSI trace $R(t_1 : t_k)$ and the best match $R_G(t'_1 : t'_l)$. Using the match for the end point of R, $a_{\min}$, we can estimate the remaining time Δt until the end of R_G as

$$\Delta t = (t_k - t_1)\frac{l - a_{\min}}{a_{\min}}.$$

We use this information to synchronously change the routing state of the whole network when necessary.

6. ROUTING

Our routing algorithm addresses the mobile user via a relay node which is part of the static network. The relay node temporarily becomes sink for all user data packets and we will use the terms sink and relay node interchangeably in the rest of this paper. We use information potential based routing [26] to deliver messages from any node in the network to the sink.

An *information potential* is a real function on the nodes that is defined for a specific node, which in our application is the relay node. It is defined to be the function that meets the following requirements: its value is (a) 1 at the sink, (b) 0 at some other node (which we ensure can never be the relay node), (c) at any other node, the function value equals the average value of its neighbors. It can be shown that there is a unique such function for any given sink (it is the harmonic function meeting the specified boundary conditions). The information potential is a 'smooth' function with no local extrema [21]. It is stable to small changes in network connectivity.

Alternative interpretation of the information potential value at some node is the probability that a random walk starting from that node will hit the sink before it hits network boundary. In particular, potential-based routing takes into account not only the length of the shortest path to the sink, but also the 'robustness' of that path, i.e. the number of alternative, perhaps slightly suboptimal paths that lead to the sink. It prefers a path to the sink which may be a little longer but follows a wide corridor, to a possibly short but thin path, which can be fragile under link volatility. We feel that this robustness property is important in wireless network setting.

The above definitions imply that forwarding a packet to the neighbor with the highest potential value, i. e. following the gradient of the potential guarantees packet delivery to the sink. More precisely, an information potential induces a *routing tree* whose edges connect each node to the neighbor with the highest potential value. For a more detailed description of information potentials and a detailed discussion of their properties in static routing, we refer to [26]. In the following, we will briefly describe how we compute the potential, before addressing our extensions regarding dynamic updates.

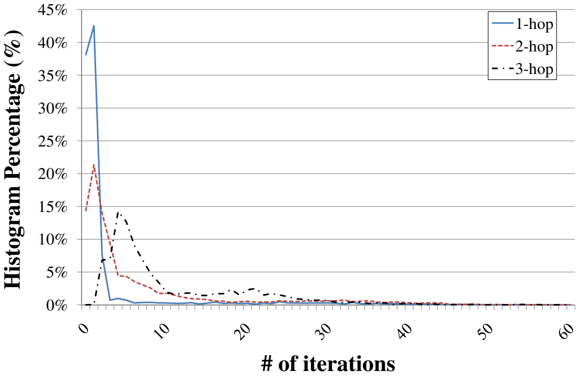


Figure 5: Convergence of routing trees induced by information potentials under local sink movement. We computed information potentials for 1000 random networks with 400 nodes each. The plot shows the number of iterations until the routing tree is valid (all nodes are reachable), after the sink has moved within its 1-hop, 2-hop, and 3-hop neighborhood.

6.1 Computation of Information Potentials

To compute and maintain information potentials, each node only has to communicate to its 1-hop neighbors, so the total per-node state is minimal. Specifically, we use the Gauss-Seidel iterative method, in which each node holds its own potential value, and periodically updates it to be the current average value of its neighbors (unless it is the sink or a fixed 0-node, in which case it does nothing). Initial values can be arbitrary, which is useful for dynamic updates, as we explain below. These updates can also be asynchronous; as long as they happen in regular enough intervals, the values will eventually converge to the true potential.

6.2 Updates for Mobile Sinks

Whenever a node moves, the information potential has to be adapted to the new sink. Most of the time, the new sink v is a 1-hop neighbor of the old one u in the communication graph. In that case, we expect only a few edges in the routing tree induced by the potential to change, mainly those near u and v (a version of this statement can be formally proved, using theory of resistive electrical networks). In particular, nodes that are far away from the mobile node can use the old routing tree with minimal performance penalty. Even if the routing tree is still changing, packets that are en route will eventually reach their destination, although they may take a short detour. Very few iterations suffice to repair the potential.

Fig. 5 illustrates our convergence claim for random networks with 400 nodes: If the sink moves to a neighboring node, we need an average of 3.8 iterations until

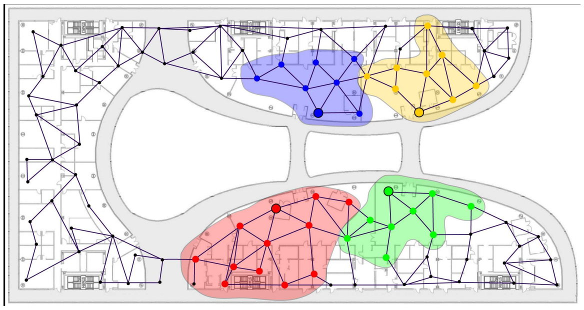


Figure 6: Potentials are precomputed at only few nodes and used by all nodes in a 2-hop neighborhood around these nodes, reducing the overall number of stored potentials.

the computed potential defines a valid routing tree for the complete network.

To further improve the routing performance, we can use mobility prediction. If we know ahead of time which will be the next relay node, we can start computing its information potential before it is active. In practice, each node stores and updates two potentials, the current potential used for routing, and the predicted potential, which is used once the transition to the predicted node took place. Thus, we start computing the new potential for a longer time, and the induced tree is likely to have converged even before the potential is needed for routing.

6.3 Precomputed Potentials

Calculating information potentials for the predicted relay node solves the mobility problem for local movement, however, we know that local movement in the real world may be non-local within the network, as in the bridge example shown in Fig. 6. We use information contained in the mobility graph to determine where and when such non-local movement can occur.

To each edge (u, v) of the mobility graph we can assign a *stretch*, which is length of the shortest network path connecting u and v. The problem of the update method described in the previous section is that for higher stretch edges, the information potential for u is not a good start value for the potential of v, and the induced routing tree will need many iterations to converge. We therefore precompute potentials for some nodes, so that in all cases, a good start value for the iteration is available. Note that we only have to precompute potentials for nodes that are the endpoints of mobility graph edges with high stretch.

To reduce the storage requirements even further, a cluster of endpoints of high stretch edges can share a common potential (see Fig. 6). When deciding where to store potentials, we apply the following criterion: if a node requires a stored potential because it is the end point of a mobility graph edge with high stretch, we first check whether a potential is stored in its k-hop neighborhood. If there is such a potential, no additional potential is required. Otherwise, the potential for this node is precomputed. This simple clustering technique leads to a very efficient sampling of the network with precomputed potentials. The number of potentials that we need to store depends on the topology of the environment, specifically, the number of large holes. In our experiments in Section 7.2, only 4 precomputed potentials are actually stored. Note also that each potential requires only one real value per node, which makes this technique extremely lightweight.

6.4 Implementation

In our system, we implement the above ideas as follows: Each node stores two active potentials (one for the current relay node and one for the predicted relay node). These potentials are continuously updated. We use a frequency of four iterations per second.

Two events change the state of the network: a new prediction is available or the relay node changes. Both events are propagated from their source (the mobile node) by network-wide flood. We piggyback this flood on the messages that periodically update the information potentials. On one hand, this limits the speed of the flood to the speed of the potential updates, but on the other hand, no extra messages are introduced. The flood contains four pieces of information: the current relay node ID, the predicted relay node ID, the precomputed potential ID (if any) and the estimated time to transition.

All these fields are provided by the mobile node (who performs the prediction), and passed on by all nodes in the network. If a node is named as current relay, it changes its own field value for the current information potential to 1 and does not change it in iterations. If a node is named as predicted relay node, it changes its predicted information potential to 1 and does not change it in iterations. Finally, all nodes set an internal timer to the estimated time to transition, such that the network switches to the new potential in a synchronous manner. Even though the accuracy of synchronization is not critical in this case, note that synchronous coordinated action in WSNs can be achieved to within a few microseconds accuracy [25].

6.5 Extensibility to Multiple Users

Straightforward extension of our routing scheme to several mobile users would require to store two potentials per user. The communication and storage requirements therefore grow linearly with the number of mobile users. Note that each potential requires only one float value of storage space. Even with tens of users, updating all potentials is possible using a single message, thus the communication cost for additional users would be

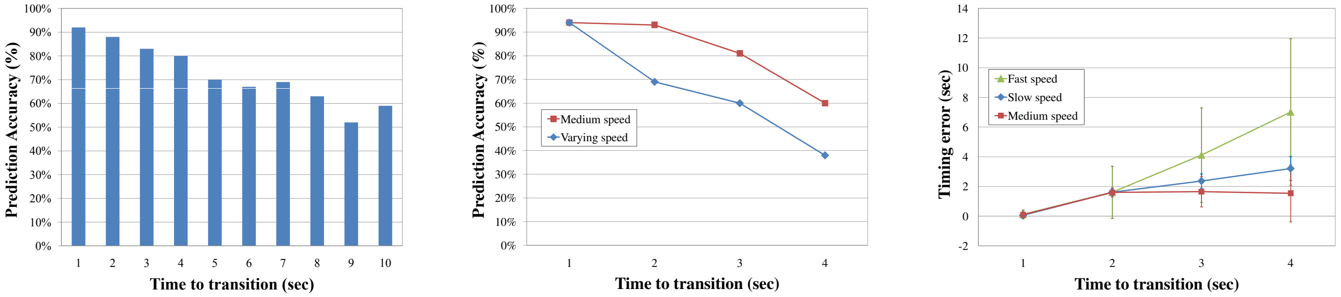


Figure 7: Experimental evaluation of prediction algorithm. Left: Mean accuracy of prediction of the next node, for varying time to transition to the node. Middle: Prediction accuracy for different speeds. ■: Training and testing speeds are similar, ◇: Training and testing speeds differ by ±30%. Right: Error in estimation of time to transition, showing the mean error and the standard deviation. ■: Training and testing speeds are similar. ◇: Slower testing speed. ▲: Faster testing speed.

negligible. Therefore, the method is practical for most sensor network deployments.

For very large networks with thousands of nodes and hundreds of users, however, it would become prohibitive to maintain potentials for every user. Moreover, as we show in the next section, information potentials do not change significantly for distant users. Consequently, hierarchical clustering techniques for users similar to [15] could be applied to reduce storage requirements.

7. EVALUATION

Firstly, we experimentally evaluate the mobility prediction algorithm described in Section 5. We have deployed a small testbed in a 750 m^2 indoor office space and tested the prediction accuracy in a series of experiments. In the second part, we evaluate the predictive routing algorithm described in Section 6. We used a map of our building to define a realistic 100 node topology in an 9000 m^2 area. We implemented an extension to TOSSIM simulator that allowed us to simulate mobility of users along predefined trajectories.

7.1 Mobility Prediction

We tested the mobility prediction algorithm experimentally, in a network of 10 MicaZ [9] motes deployed in an office space (see Fig. 2a). We have covered the area of approximately 30 m × 25 m, leading to a 3 hop network. The infrastructure nodes were programmed in TinyOS-2.1 [16] and recorded time-stamped RSSI of messages received from a mobile mote. The mobile node broadcasted messages with a 0.6 seconds period with a constant transmission power throughout all experiments. All experiments were taken during regular working hours, with people and equipment moving around, doors opening and closing, and with the mobile node moving both inside and outside of the building.

The algorithm requires a set of observation sequences to learn the mobility graph. Altogether, we have collected data for 9 different trajectories, repeating each of them at least 5 times. We selected 5 of these trajectories for learning and used the remaining 4 for testing. None of the testing trajectories was the same as any learning trajectory, although learning and testing trajectories overlapped in some segments. The trajectories were collected over a one week period to account for variance of RSSI signals over time.

The prediction algorithm needs to be able to reliably predict the next relay node sufficiently in advance, to leave enough time for the routing protocol to seamlessly adapt to the new relay node. In general, prediction of a few seconds ahead is sufficient.

We first evaluate accuracy of the prediction algorithm: for a given observation sequence $R(t_1 : t_k)$, we construct the best neighbor sequence $B(t_1 : t_k)$ to obtain the ground truth of which node is the relay node throughout the experiment. In the online phase, we predict the next relay node and estimate the time to transition to the new relay node, Δt. We initially use only the first observation $R(t_1)$ for prediction and incrementally consider more observations $R(t_1 : t)$, until $t = t_k$. For each prediction, we compare the predicted relay node and the predicted time to transition to the ground truth and calculate the ratio of correct predictions relay nodes. We show the histogram of prediction accuracy depending on the actual time to transition in Fig. 7 (a). As we can see in the figure, the accuracy of the prediction is initially low, since only few data points are available for prediction. As the number of data points grows (with decreasing time to transition), the accuracy of the prediction improves significantly, ending around 90% for prediction 1-2 seconds before the transition.

We further test how well the DTW algorithm can compensate for speed differences. Fig. 7 (b) shows the prediction accuracy of the same trajectory when the walking speed in training and testing samples were identical, and when we changed the walking speed in the testing phase by ±30%. Although the prediction accu-

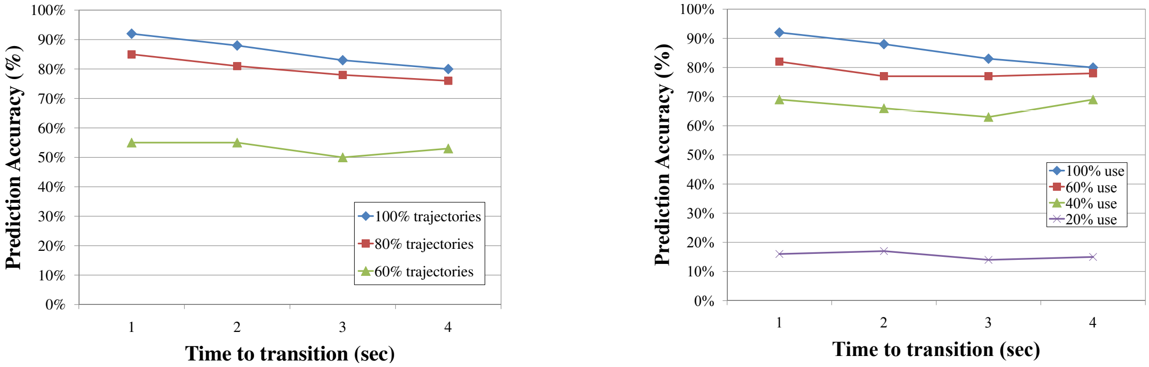


Figure 8: Impact of size and variability of training data. Left: Prediction accuracy depending on the size of the training set. Right: Prediction accuracy depending on the variability of the training set.

racy suffers, we will see that this has little impact on the routing performance.

Also, we measure the error in the estimation of time to transition. Fig. 7 (c) shows the mean error in the estimate, along with the standard deviation of the error. We will use this data below to build a statistical prediction model for the routing simulation.

Finally, we explore the impact of the size and variability of the training set on prediction accuracy. First, we discard datasets of some training trajectories (out of the five trajectories we used in total). Fig. 8 (a) shows that the prediction accuracy degrades significantly as we remove trajectories from the training set. The main reason is that the remaining training trajectories capture mobile patterns only partially and it is difficult to predict trajectories that have never been observed. Next, we kept all training trajectories, but removed some of the repetitive training rounds for each trajectory. Fig. 8 (b) shows that the prediction accuracy is less dependent on the variability of the training set. Even when removing 60% of the training data, the prediction accuracy is quite high. Overall, this evaluation implies that exploring the complete routes over the network with fairly many training runs is an important factor to achieve higher prediction accuracy.

7.2 Predictive Routing Simulation

We simulated a larger scale 9000 m^2 setup covering one floor of our building to test performance of the routing algorithm. Due to the layout of the building, the network is U-shaped. Hence, the mobility graph contains edges with high stretch. We implemented the routing algorithm in TOSSIM as described in Section 6 and ran experiments for 2 different paths, 6 different speeds, and 10 different error models. For a fixed path, speed, and error model, we simulated 1000 transmitted packets overall and repeated each experiment 5 times to obtain statistically significant data. Altogether, over 300 simulations were conducted to evaluate performance of the routing algorithm under different scenarios.

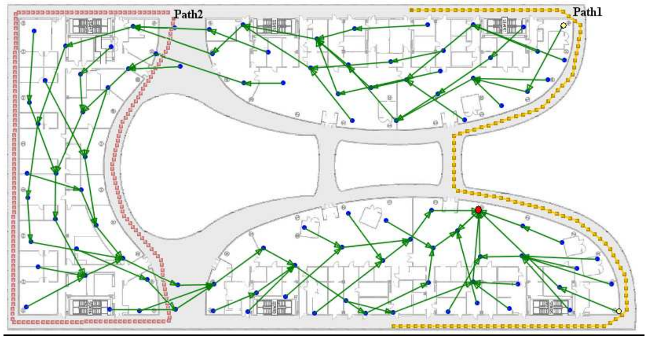


Figure 9: TOSSIM deployment setup and an example of a routing tree obtained from information potentials. Infrastructure nodes are shown as blue dots, the current relay node (the root of the routing tree) is highlighted. The mobile node moves along Path 1 and Path 2.

To provide bounds on routing performance, we first evaluate perfect prediction routing (using the known trajectory of the mobile user) and then disable some components of the algorithm. This information is summarized in Fig. 10. We have tested two trajectories: Path 1 containing non-local transition in the mobility graph, and Path 2 that only contains local transitions. For Path 1, using prediction and routing optimizations improves the overall routing performance by about 50% for the highest speed. Performance of the routing algorithm for Path 2 was also above 90% for all speeds. However, the baseline algorithm that uses no prediction performed above 86% for all speeds, corresponding to a modest improvement of about 4%. We do not show the plots due to space constraints.

We have not implemented the actual prediction algorithm in TOSSIM because it is hard to generate realistic RSSI traces of mobile users in indoor environments. Hence we used a statistical model of the qualitative prediction behavior in the simulation. We use the predic-

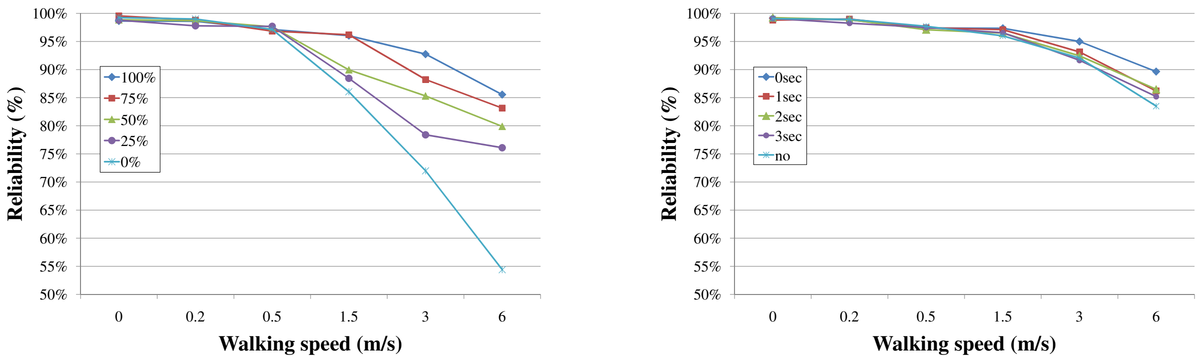


Figure 11: Routing performance for Path 1 with erroneous prediction. Left: Influence of prediction accuracy A, right: Influence of timing error τ on routing performance.

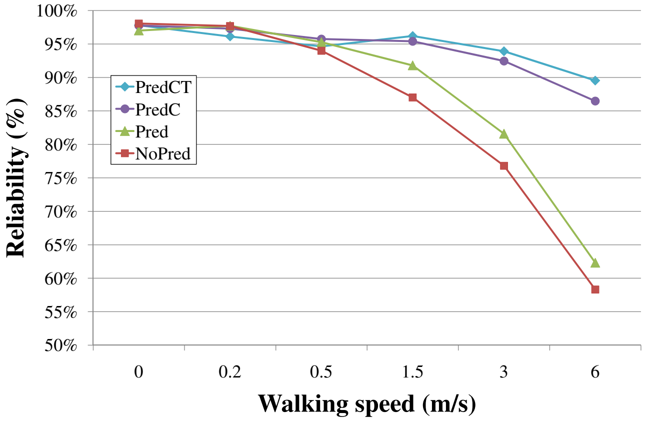


Figure 10: Path 1 routing performance. ■: no prediction. ▲: perfect prediction of the next relay node but no precomputed information potentials. ●: perfect next relay prediction and precomputed potentials, ◇: perfect prediction of both the next relay node and the estimated time to transition, and precomputed potentials.

tion accuracy A, mean timing error τ, and the standard deviation of the timing error σ to build the model.

7.2.1 *Performance Under Failure*

For the prediction of the relay node, we draw a random number $0 \leq \eta \leq 1$. If $\eta < A$, we predict the correct next relay node, otherwise, we chose a random (but incorrect) outgoing edge incident to the current node from the mobility graph, and predict its end point as the next relay node. Fig. 11 (a) shows the effect of A on the routing performance. In case of the timing error, we assume it is drawn from a normal distribution with mean τ and standard deviation σ. Fig. 11 (b) shows the effect of different values of τ. The graphs show performance for Path 1 only. In both cases, we can see that the quality of service achieved using our method is significantly higher than the base case, even if the quality of prediction is unrealistically low.

Fig. 11 also shows an important property of the routing implementation: its performance gracefully degrades to the original routing algorithm in the case of wrong (or non-existent) prediction. This is because if the predicted node is different from the actual node, we discard the predicted gradient value and adapt the existing gradient field to the new relay node.

7.2.2 *Convergence of the Routing Tree*

It is interesting to note that the routing tree first converges at the nodes close to the mobile user. This is due to two reasons: the broadcast carrying new location of the user is piggybacked on the potential field update messages, and thus reaches faraway nodes slowly. Additionally, the area around the relay node is close to a boundary condition for the potential field (the relay node has a fixed value of 1), and therefore converges faster. Slower convergence at faraway nodes is not a problem as the gradient direction at those nodes does not depend on the accurate location of the user.

8. CONCLUSION

We have presented a sensor network data delivery protocol for mobile nodes. Our algorithm allows all nodes in the network to send data to one (or several) mobile nodes. We achieve gains in routing performance using novel components: a mobility prediction algorithm, and a variant of gradient-based routing that is able to adapt to a moving user, including the use of precomputed information potentials for non-local movements.

We heavily build on the concept of the mobility graph, a data structure that encapsulates and formalizes knowledge about possible mobility patterns of users roaming the sensor network. Especially in structured, man-made environments, the mobility graph is a valuable tool for understanding and optimizing wireless networks.

In the future, we aim to show the scalability of the approach to very large networks. To prevent explosion of the routing state at each node, we need to limit the number of precomputed potentials each node stores. Since the potentials need to be precomputed only for large features in the environment, and since nodes do not need to know about far away potentials, each node should be limited to store only a few precomputed values. We would also like to explore techniques that would allow multiple users to share the same potential field, without significantly increasing routing cost.

Finally, we would like to improve communication cost of our protocol. We have shown that very few iterations are required to update the potential field and that the field changes significantly only close to the mobile node. Consequently, a Trickle timer that exponentially increases its period when the potential field is stable could significantly decrease the cost of the field update.

9. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of Army AHPCRC grants W911NF-07-2-0027-1, W911NF-06-1-0275 and NSF grants CNS-0626151, CCF-0634803, and of the Max Planck Center for Visual Computing. HyungJune Lee is funded by a Samsung Scholarship.

10. REFERENCES

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *DIALM-POMC '04: Proc. Joint Workshop on Found. of Mob. Comp.*, pages 75–84, 2004.

[2] B. Awerbuch and D. Peleg. Online tracking of mobile users. *J. ACM*, 42(5):1021–1058, 1995.

[3] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *Int. J. Rob. Res.*, 24(1):31–48, 2005.

[4] A. Bhattacharya and S. K. Das. Lezi-update: an information-theoretic framework for personal mobility tracking in pcs networks. *Wireless Networks*, 8(2/3):121–135, 2002.

[5] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proc. IPSN '07*, pages 450–459, 2007.

[6] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet-Draft, draft-ietf-manet-dymo-12.txt, 2008.

[7] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR), 2003.

[8] L. J. Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

[9] Crossbow Technology Inc., http://www.xbow.com. *MICAz wireless measurement system*, June 2004.

[10] M. Demirbas, A. Arora, and V. Kulathumani. Glance: A lightweight querying service for wireless sensor networks. In A. A. Shvartsman, editor, *OPODIS*, volume 4305 of *Lecture Notes in Computer Science*, pages 244–259. Springer, 2006.

[11] M. Demirbas, A. Arora, T. Nolte, and N. Lynch. A hierarchy-based fault-local stabilizing algorithm for tracking in sensor networks. In *Principles of Distributed Systems*, pages 299–315. Springer Berlin / Heidelberg, 2005.

[12] M. Demirbas, O. Soysal, and A. S. Tosun. Data salmon: A greedy mobile basestation protocol for efficient data collection in wireless sensor networks. In *Proc. IEEE Int. Conf. on Dist. Comp. in Sensor Sys.*, 2007.

[13] F. Feng and D. Reeves. Explicit proactive handoff with motion prediction for mobile ip. In *Proc. of the Wireless Comm. and Networking Conf.*, 2004.

[14] J. Froehlich and J. Krumm. Route prediction from trip observations. *Society of Automotive Engineers (SAE) 2008 World Congress*, 2008.

[15] S. Funke, L. Guibas, A. Nguyen, and Y. Wang. Distance-sensitive information brokerage in sensor networks. *Proc. Int. Conf. Dist. Comp. in Sensor Systems (DCOSS)*, pages 234–251, 2006.

[16] D. Gay, P. Levis, and D. Culler. Software design patterns for tinyos. *Trans. on Embedded Computing Sys.*, 6(4):22, 2007.

[17] K. Hwang, J. In, and D. Eom. Distributed dynamic shared tree for minimum energy data aggregation of multiple mobile sinks in wireless sensor networks. In *Proc. EWSN*, 2006.

[18] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mob. Netw. Appl.*, 11(3):327–339, 2006.

[19] D. Jea, A. Somasundra, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Proc. IEEE Int. Conf. on Dist. Comp. in Sensor Sys.*, 2005.

[20] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proc. MobiCom '02*, 2002.

[21] S. G. Krantz. *Handbook of Complex Variables.* Birkhauser, Boston, MA, 1999.

[22] J. Krumm. A markov model for driver route prediction. *Society of Automotive Engineers (SAE) 2008 World Congress*, 2008.

[23] J. Kruskall and M. Liberman. The symmetric time warping problem: From continuous to discrete. In *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 125–161. Addison-Wesley Publishing Co., 1983.

[24] V. Kulathumani, A. Arora, M. Demirbas, and M. Sridharan. Trail: A distance sensitive wsn service for distributed object tracking. In K. Langendoen and T. Voigt, editors, *EWSN*, volume 4373 of *Lecture Notes in Computer Science*, pages 83–100. Springer, 2007.

[25] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services. *Int. J. of Ad Hoc and Ubiquitous Comp.*, 2(1), 2006.

[26] H. Lin, M. Lu, N. Milosavljević, J. Gao, and L. J. Guibas. Composable information gradients in wireless sensor networks. In *Proc. IPSN '08*, pages 121–132, April 2008.

[27] Liu and Maguire. A class of mobile motion prediction algorithms for wireless mobile computing and communications. *Mobile Networks and Applications*, 1996.

[28] Y. Mao, F. Wang, L. Qiu, , S. Lam, and J. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the Fourth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[29] C. Papamanthou, F. P. Preparata, and R. Tamassia. Efficient localization for wireless sensor networks using power measurements sampling.

[30] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. IETF Internet draft, draft-ietf-manet-aodv-09.txt, November 2001 (Work in Progress).

[31] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, 1994.

[32] L. Song, D. Kotz, and X. He. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Trans. on Mob. Comp.*, 5(12):1633–1649, 2006.

[33] TEP 123. Collection Tree Protocol. http://www.tinyos.net/tinyos-2.x/doc/.

[34] M. Thorup and U. Zwick. Compact routing schemes. In *SPAA '01: Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, pages 1–10, 2001.

[35] D. Vasquez and T. Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. ICRA*, 2004.

[36] W. Wang, V. Srinivasan, and K. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proc. MobiCom*, 2005.

[37] Widyawan, M. Klepal, and D. Pesch. Influence of predicted and measured fingerprint on the accuracy of RSSI-based indoor location systems. In *4th Workshop on Positioning, Navigation and Communication (WPNC '07)*, 2007.

[38] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networks. In *Proc. Conf. on Emb. Networked Sensor Sys.*, Los Angeles, CA, Nov. 2003.