Data Stashing: Energy-Efficient Information Delivery to Mobile Sinks through Trajectory Prediction

HyungJune Lee[1], Martin Wicke[2], Branislav Kusy[3],
Omprakash Gnawali[1], and Leonidas Guibas[1]
[1]Stanford University, Stanford, CA, USA
[2]University of California, Berkeley, CA, USA
[3]CSIRO ICT Centre, Brisbane, Australia
abbado@stanford.edu, wicke@eecs.berkeley.edu, branislav.kusy@gmail.com
{gnawali,guibas}@cs.stanford.edu

ABSTRACT

In this paper, we present a routing scheme that exploits knowledge about the behavior of mobile sinks within a network of data sources to minimize energy consumption and network congestion. For delay-tolerant network applications, we propose to route data not to the sink directly, but to send it instead to a relay node along an announced or predicted path of the mobile node that is close to the data source. The relay node will *stash* the information until the mobile node passes by and picks up the data. We use linear programming to find optimal relay nodes that minimize the number of necessary transmissions while guaranteeing robustness against link and node failures, as well as trajectory uncertainty.

We show that this technique can drastically reduce the number of transmissions necessary to deliver data to mobile sinks. We derive mobility and association models from real-world data traces and evaluate our data stashing technique in simulations. We examine the influence of uncertainty in the trajectory prediction on the performance and robustness of the routing scheme.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer- Communication Networks; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Algorithms, Design, Measurement, Performance

Keywords

Mobile Data Delivery, Trajectory Prediction, Mobility Pattern, Network Optimization, Sensor Networks

IPSN'10, April 12–16, 2010, Stockholm, Sweden.

1. INTRODUCTION

Classic multi-hop wireless routing protocols compute the shortest path (in hops or metrics such as ETX [8]) between sources and destinations in a network. Since the shortest path minimizes the number of necessary transmissions, this strategy minimizes not only delay but also energy use.

In the presence of mobility, however, the shortest path computed at one point in time is not necessarily the shortest possible path connecting the source and the sink. A shorter path might be available, if the nodes move closer to each other in the future. An optimal routing strategy can be devised if the trajectory of the mobile nodes is known.

In this paper, we study the problem of sending information from the nodes in a sensor network to multiple mobile sinks moving in the same space as the network. Given some information about each sink's trajectory, we aim to minimize the expected routing cost to the sink. We assume that the information sources and sensor network nodes are static (not mobile), while data sinks (humans or vehicles) move inside the area covered by the sensor network. Finally, we assume that applications tolerate a packet delivery delay in the order of the average network traversal time for mobile nodes, e.g., a few minutes. This is often the case in sensor networks that accumulate measurements until an observer takes a reading [34]. Examples of such data delivery patterns can also be found in applications that sense information in places where people work or live and deliver it to user mobile devices, enabling more intelligent living environments.

We solve the problem of data delivery to multiple mobile sinks by *stashing* data along the anticipated trajectory of the sink, instead of routing the data directly to the sink at its current position. If we can accurately predict that the mobile sinks pass near a particular sensor node at some point in time, we can stash data on this *stashing node*. As each mobile sink passes this node, it can pick up the data.

We use linear programming (LP) to find, for each data source, the optimal stashing node to which to send the data. We develop this optimization to take into account multiple mobile sinks requesting the same information, as well as considerations about robustness in the presence of uncertainty about the trajectories of the mobile sinks. Our linear programming formulation requires some information about the future trajectory of mobile sinks. In some applications, sinks know their future trajectory through the network and can announce it to the network when requesting information. Even if the future trajectory is unknown, many applications

are deployed in environments that constrain motion patterns of sinks to roads, trails, or hallways. Our algorithm aims to exploit such structure without explicitly representing it.

We predict the motions of mobile sinks by using association data and a history of trajectories. We present a method for representing trajectories, learning typical trajectories from observations, as well as predicting likely trajectories given observed partial trajectories. The prediction algorithm can be used by the mobile sink (or its closest relay node) to supply information about its future trajectory to the network. We characterize the trajectories as sequences of node associations, and use multiple sequence alignment techniques to compute similarity on partial sequences. Using this similarity metric, we compute clusters representing typical trajectories through the network. For efficiency, we find a compact probabilistic representation for the clusters which we use to efficiently find likely future trajectories during prediction.

We evaluate our probabilistic trajectory model used for prediction on data taken from the DieselNet traces [4]. We use simulation experiments to validate our approach by comparing our technique against direct routing in terms of routing efficiency and robustness. We show that one benefit of our technique is better load balancing and more even utilization of network resources, such as energy.

In summary, our contributions are the following:

- We present *data stashing*, a data delivery scheme that routes data to mobile sinks, but lets each sensor node decide where on a set of possible trajectories it wants to stash its data, to be picked up whenever the mobile sink passes the stashing node.
- We introduce a network-centric representation for trajectories. In this representation, a trajectory is represented as a sequence of associated nodes, giving us all the information we need for data delivery, while abstracting from unnecessary and possibly misleading spatial information. We also develop useful similarity measures for this motion representation which allows us to perform clustering.
- We propose a probabilistic representation for sets of similar (but potentially partial) trajectories. This representation can be used to compactly describe a cluster of trajectories, and efficiently find the best-matching cluster given a partial trajectory.

2. RELATED WORK

There is a large body of research in routing protocols designed to deliver packets to mobile sinks in wireless networks. Some of these protocols assume little about the network and the mobility pattern of the mobile sinks and perform network discovery pro-actively or on-demand. Classic protocols such as DSR [14] and AODV [33], which were originally designed for wireless ad hoc networks, and sometimes used in mobile routing, fall into this category. In the wireless sensor network context, protocols such as SEAD [16] and TTDD [41] construct energy-efficient routing paths without knowledge of the mobility patterns of the sink. However, in some scenarios, the mobile sinks move in a pattern that can be predicted to a certain extent. This observation inspires the second category of routing research, in which we make assumptions about the trajectory of the mobile sink. The routing protocols can then exploit these patterns to efficiently deliver packets to the mobile sink. This category of routing protocols, of which ours is one example, typically consists of two components: mobility pattern analysis/learning followed by path computation and packet delivery.

Mobility patterns have been studied using GPS data, or association data from cellular networks or wireless LANs. In the case of GPS, since the raw GPS data contain many outliers, most of the previous research approaches [3, 9, 18] filter out the noisy and unreasonable measurements first, and then identify the possible goal locations from the filtered GPS positions, and construct prediction models. Ashbrook and Starner [3] find significant places where a user spent over a threshold amount of time, and cluster them into locations with the k-means clustering algorithm. Finally, a Markov model is applied for each location, and used for predicting the next goal location. Froehlich and Krumm [9, 18] obtain the end-to-end routes from the raw GPS data, and use a Bayesian model and a trip similarity clustering algorithm to predict the next goal location. Further, Liao et al. [25, 26] and Yin et al. [42] not only extract significant places from filtered GPS data, but also try to associate the places with activities that a person can undertake in each different place. Their work is the first to suggest exploiting high-level context (i.e., user's activities) to detect the goal place for a mobile user with higher fidelity. Similarly, in cellular networks, some previous work [5, 20, 21, 32] uses cell identifiers to identify significant locations, and constructs prediction models by clustering algorithms. In wireless LAN networks, a long-term large-scale measurement study of user-access point (AP) association at Dartmouth [17] has inspired work in mobility prediction. It has been noted that wireless users' locations can be predicted with up to 72% accuracy using an order-2 Markov predictor [37] for users with long trace lengths. Further analysis of the same dataset has suggested the feasibility of predicting the trajectory of a mobile user in space and time [36]. Using a different dataset, Ghosh et al. [11] describe techniques to predict a user's location with respect to social hubs such as buildings and classrooms, rather than individual wireless AP's.

There has been previous work on exploiting predicted mobility to improve the efficiency of routing to sinks with predictable trajectories. Our previous work on *mobility graphs* allows the network to predict future relay nodes [19]. Information potentials for the predicted nodes are computed while the old route is still valid, enabling an instantaneous switch to the new relay node. Chakrabarti et al. [6] proposed a protocol in which the sensor nodes keep statistics of sink visits and transmit information only when the mobile sink is within transmission range. Our work does not assume that the trajectory of a mobile node takes it within single-hop transmission range of each sensor in the network. Most closely related to our work is recent work on the proactive scheme in TwinRoute [40]. Based on the sink arrival statistics, a subset of nodes elect themselves as storage nodes and initiate routing tree construction as roots. The sensor network forwards data to these storage nodes so that packets can be relayed to the mobile sink. Although our work fits in this general framework, we employ different methods to overcome shortcomings of this approach. We use clustering to improve the accuracy of trajectory prediction. We

compute routes that are globally optimal, while explicitly accounting for multiple mobile sinks. Our objective is to deliver packets to the sink with the least network overhead, while assuming no data freshness requirement.

There is a large body of work on routing to mobile sinks with trajectories that can be programmed to optimize data forwarding efficiency [10, 28, 29]. Our work does not assume a programmable trajectory of the mobile sinks. Researchers have also formulated computing energy-efficient routes in sensor networks as an optimization problem [7, 24, 27]. Our work also frames routing as an optimization problem. However, in our LP formulation a number of stashing nodes or the sinks themselves can be feasible destinations, while also taking into account link reliability and the probabilistic nature of the predicted trajectories of the mobile sinks.

3. OVERVIEW

The main objective of this paper is to develop a routing scheme that delivers data to mobile sinks through a wireless mesh sensor network. We exploit knowledge about the mobility of the sinks to lower the cost and increase the reliability of data transmission. We will use the terms mobile sink and mobile node interchangeably in the rest of this paper.

In particular, we solve the following problem: One (or several) mobile sink moves through a network, collecting sensor data from nodes in the network. Traditionally, we would either send all data directly to the current position of the mobile sink (that is, to a node that is close to the mobile sink, which will relay the information to the mobile sink), or not send any data at all, and wait for the mobile sink to collect the data as it passes each of the sensor nodes. The latter option is often infeasible if we cannot control the movement of the mobile node, or if moving within radio range of each desired sensor is not an option. We choose a compromise between the two extremes. Using knowledge about the trajectory of the mobile node, sensors route data to a set of *stashing nodes* that store information along the likely trajectories of the mobile node.

At the core of our method is an optimization procedure that for each sensor chooses a set of stashing nodes that guarantee (with high probability) that a mobile node will receive the sensor's data. See Fig. 1 for an illustration. The optimization procedure is described in detail in Sec. 6.

We assume some knowledge about the possible trajectories that a mobile node can take. This information either comes from the mobile node itself, or is deduced from observations of motion patterns of sinks in the network. We represent trajectories as strings of associated mesh network nodes. We define a distance on the space of trajectories and use clustering to find representative trajectories. Algorithms for trajectory representation and clustering are described in Sec. 4, and prediction is in Sec. 5.

Our evaluation in Sec. 7 shows that exploiting knowledge of sinks' motion can greatly decrease transmission costs and energy use. However, we do require sensor nodes to have some storage capacity, and we assume that the data delivered to the mobile sink is delay-tolerant. The sink will collect the data throughout its journey through the network, possibly introducing some delay in data availability to the sink.

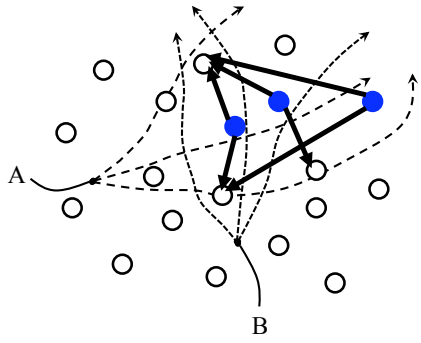


Figure 1: Given a set of trajectories, we select an optimal set of stashing nodes for each sensor node. This set minimizes transmission cost, but ensures that no matter which trajectory is used by each mobile node, the data will be collected.

3.1 Protocol

In order to clarify the process of trajectory prediction, stashing node selection, and routing, we give a high-level description of the protocol used to negotiate data stashing for a mobile sink. The protocol assumes that a mobile sink enters the network and requests data from a set of sensor nodes.

1. **Trajectory prediction.** When a mobile sink joins the network, it beacons in regular intervals. Sensor nodes in range reply with their IDs and the sink selects the node whose reply was received with the strongest signal as its *relay node*. As the sink moves through the network, this yields a string of relay node IDs. We use this string to predict a set of likely trajectories that most closely match the recorded prefix in the database of historical trajectories acquired in an off-line learning phase, as described in Sec. 5. If the trajectory or set of likely trajectories is known, this step can be skipped.

2. **Data request and trajectory announcement.** The mobile sink announces the set of likely trajectories to the network. The set of trajectories is encoded and broadcast to the whole network. This message can also contain a set of sensor nodes whose data are interesting to the mobile node.

3. **Stashing node selection.** Upon receiving a sink's request for data and a set of likely trajectories, each sensor node computes a set of stashing nodes that cover the likely trajectories and minimize the routing cost required to send the data to the stashing nodes. The optimization procedure is described in Sec. 6.

4. **Data stashing.** Sensor nodes forward data to the stashing nodes, for future delivery to mobile sinks.

5. **Data collection.** As the mobile node moves through the network, it regularly beacons to announce its position. If a stashing node receives a beacon, it starts transmitting the data stashed at this node to the mobile node.

This protocol is easily extensible to multiple mobile sinks. We disambiguate between the sinks based on their unique IDs and discuss scenarios with multiple mobile sinks in Sec. 7.

Note that we assume an underlying point-to-point routing protocol such as S4 [30], however, we make no assumptions on the properties of this protocol.

In the following sections, we present the components of our system. First, we discuss how we represent trajectories, including a similarity measure on trajectories that allows us to meaningfully cluster trajectories in Sec. 4. In Sec. 5, we describe how we predict trajectories using a database of recorded trajectories (step 1 in Sec. 3.1). Finally, we present the optimization process for selecting stashing nodes (step 3 in Sec. 3.1) in Sec. 6, before we evaluate results in Sec. 7.

4. TRAJECTORIES AND CLUSTERS

In most scenarios, mobile sinks travel along a fairly limited set of trajectories. Oftentimes, this is due to obstacles present in the environment: buildings, bridges, roads, and walkways constrain the possible trajectories. Even without any environmental restrictions, there are usually few interesting start- and endpoints for any given journey, and sinks often follow short(-est) paths, greatly limiting the set of possible trajectories.

It therefore makes sense to find and exploit the structure that is present in the likely trajectories through a network. We will do so by clustering similar trajectories, thus creating a database of historical trajectories, arranged in clusters of similar trajectories in the off-line learning phase. In order to perform practical clustering on trajectories, we require a trajectory representation, a similarity measure, and a compact representation of a cluster of sequences. The following sections describe these concepts in turn.

4.1 Trajectory Representation

In the following, we will represent a single trajectory through the network not in terms of spatial position, but in terms of the best-connected sensor node at any given time.

Let us consider a mobile sink moving through the network on a given spatial trajectory. Sending periodic beacons and listening for replies, the mobile node can record the nodes in radio range at each beacon time. In each of these sets, we can determine the best-connected node, for example, by measuring signal strength on the acknowledgment or the beacon packet. This is the node that the mobile node would associate with to send or receive data. We represent trajectories through the network as a sequence of best-connected nodes:

$$T = N_1 N_2 N_3 \dots N_k.$$

We only record changes in the best-connected node, i. e. $N_i \neq N_{i+1}$. For example, given "s s a a a r r r a n n g h h h h a a e e e e y y o o", the corresponding trajectory is represented as $T =$ s a r a n g h a e y o.

Note that due to imperfect links and radio signal strength fluctuations in dynamic environments, two node sequences recorded from the same spatial trajectory are not necessarily identical, or even of the same length. To compensate for noisy fluctuations in capturing similar trajectory patterns, we borrow a similarity measure from computational biology where functional, structural, or evolutionary relationships between sequences encoding biological macromolecules have been thoroughly investigated.

4.2 Similarity Measure

We use a variant of the *longest common subsequence* metric known from string theory and a variant of the *Smith-Waterman* algorithm [35] to calculate this similarity measure between two sequences.

Informally, to compute the similarity between two sequences $T_A = A_1 \dots A_{n_A}$ and $T_B = B_1 \dots B_{n_B}$, we count how many nodes we have to *insert*, *delete*, or *substitute* in T_A to obtain T_B.

We define the partial match function $F_{AB}(i, j)$, which computes the similarity between the prefixes of length i and j of T_A and T_B, $A_1 \dots A_i$ and $B_1 \dots B_j$. F_{AB} can be defined recursively:

$$\begin{aligned} F_{AB}(i,0) &= 0 \quad \text{for} \quad 0 \le i \le n_A, \\ F_{AB}(0,j) &= 0 \quad \text{for} \quad 0 \le j \le n_B, \\ F_{AB}(i,j) &= \max\big[F_{AB}(i-1,j-1) + s(A_i,B_j), \\ &\qquad F_{AB}(i-1,j) + d, \\ &\qquad F_{AB}(i,j-1) + d, \\ &\qquad 0\,\big], \end{aligned} \tag{1}$$

where the similarity for insertion or deletion operations, d, as well as the similarity function on individual nodes are free parameters. In our experiments, we use $d = 0$, meaning we see no similarity in deletion or insertion operations, and we set $s(A, A) = 1$ and $s(A, B) = 0 \ \ \forall A \neq B$. With these parameters, $F_{AB}(n_A, n_B)$ is the length of the *longest common subsequence* in the two sequences.

We often need to compare several partial trajectories A to a significantly longer complete trajectory B. As it is defined above, $F_{AB}(n_A, n_B)$ will be lower the shorter A is, even if (in the matching part of B) there is a perfect match. To compensate for differences in length of A or B, we normalize the similarity measure by dividing by the length of the shorter sequence:

$$S(A,B) = \frac{F_{AB}(n_A, n_B)}{\min(n_A, n_B)}.$$

4.3 Cluster Representation

Based on the pairwise similarities between all pairs of sequences, we apply a hierarchical clustering method for classifying each mobility trajectory into a certain number of characteristic mobility pattern clusters. We use the *average linkage* metric which uses the average similarity between objects in two clusters to determine whether clusters are merged. For a more detailed description of the hierarchical clustering method, we refer to [15].

Each cluster consists of a number of similar sequences. During the prediction stage of our algorithm, we will be presented with a partial trajectory T and asked to find the most likely cluster for this trajectory. While it would be possible to compute average linkage for T and each cluster, this would entail computing the similarity between T and each trajectory in the database. To avoid limiting the size of our database, we instead propose a probabilistic representation for each cluster, so that we can efficiently query for the best matching cluster.

We create a representation for our clusters in two steps: for each cluster, we first align all its sequences and then create a probabilistic summary of the aligned sequences.

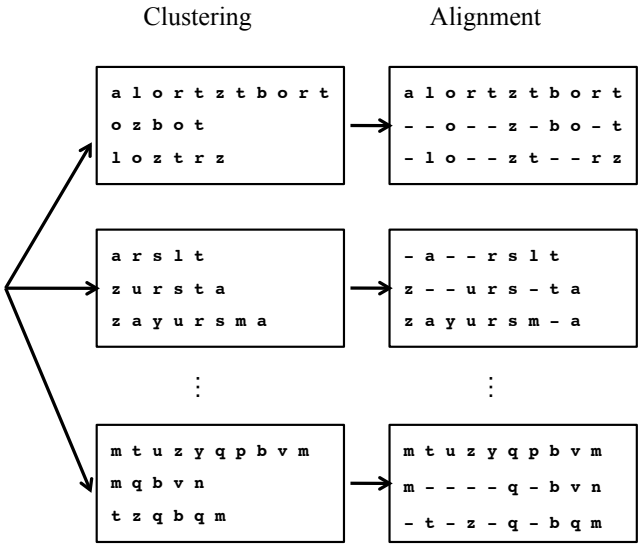


Figure 2: Clustering and alignment procedures.

4.3.1 Multiple Sequence Alignment

Given a set of sequences, multiple sequence alignment algorithms compute how the sequences should be lined up in order to maximize overlap. Our algorithm for computing the similarity between two sequences essentially computes a sequence alignment for these two sequences. In the general case, however, multiple sequence alignment is an NP-hard problem [39]. Heuristic alignment methods are widely used for DNA or protein alignments in bioinformatics [31]. We use a modified version of ClustalW, one of the most popular alignment tools [38].

The ClustalW algorithm starts by aligning the most similar sequences, and progressively adds more distant sequences one by one. This iterative procedure yields a good alignment of all sequences. We have changed the alphabet of twenty amino acids or four DNA base pairs used in computational biology to the set of node IDs more suitable for our situation. We also use an unweighted substitution matrix, making each substitution equally likely. The computation complexity of ClustalW algorithm is $O(N^2L^2)$ where N is the number of sequences and L is the sequence length [2]. To construct a cluster profile database, the aligned trajectory sequences need to be stored with storage cost $O(NL)$.

The output of the algorithm is aligned sequences that have the same length. Gaps in the aligned sequences are marked with a special gap symbol (see Fig. 2). We compute a probabilistic representation from these aligned sequences within a cluster.

4.3.2 Probabilistic Cluster Representation

Given the set of aligned sequences of length n, we construct a probabilistic representation for the cluster, which we call the cluster *profile*. A profile is a sequence of probability distributions $P = P_1 \dots P_n$. At each position i, the probability distribution $P_i(A)$ denotes the probability that node A appears in position i. This representation can also be considered a 0^{th} order Markov model of the set of aligned sequences.

The cluster profiles allow us to efficiently find the most likely cluster given a partial test sequence. See Fig. 2 for an illustration of clustering and alignment for profile generation, and Fig. 3 for a profile example.

5. MOBILITY PREDICTION

If the future trajectory of a mobile sink is unknown, our system tries to predict its behavior by comparing it to historical data. We show that even limited information about the future relay nodes can significantly improve routing performance in terms of transmission cost and load balancing.

Specifically, we are given a partial trajectory $T_M = N_1 \dots N_{n_M}$ recorded after the mobile sink enters the network. We would like to compute a set of trajectories through the network that are likely continuations of the recorded partial trajectory. In our experiments, we compute the cluster that T_M most likely belongs to, and use all elements in that cluster as our set of likely trajectories. For each of the returned sequences, we have to find the most likely position of the last node of our partial trajectory T_M, so that we can avoid stashing data to nodes that have already been visited by the mobile node. In the next two sections, we describe how we compute the closest cluster (Sec. 5.1), and how we compute the current position of the mobile node within the returned set of sequences (Sec. 5.2).

5.1 Cluster matching

Computing the similarity between a trajectory and a probabilistic trajectory profile is very similar to computing the similarity between two trajectories. In fact, the recursive definition (1) can be used unaltered, except that the partial match function F_{TP} now operates on a trajectory $T = N_1 \dots N_{n_T}$ and a profile $P = P_1 \dots P_{n_P}$. We need to change the definition of the per-node similarity function $s(N_i, P_j)$ to reflect the likelihood of N_i given the probability distribution P_j. We choose

$$s(N_i, P_j) = \begin{cases} eP_j(N_i) & P_j(N_i) > 0, \\ f & \text{otherwise.} \end{cases}$$

with the parameter values of $d = -1$, $e = 8$, and $f = -1$ which have proven effective in our setting. The parameters in the *Smith-Waterman* algorithm can be tuned to the problem, e. g., denser deployments incur higher variability of relay nodes, thus the parameters need to allow for additional mismatches and insertions/deletions.

5.2 Alignment

Once we have found the best-matching cluster, we need to align the partial trajectory with the sequences in the cluster in order to find the part of the trajectories that will be visited by the mobile node. All sequences in the cluster are aligned to each other and the cluster profile using multiple sequence alignment as described in Sec. 4.3.1. It is therefore sufficient to find an alignment of the partial trajectory T to the profile P. In particular, we are interested in the position J that the last node in the partial trajectory, N_{n_T}, is matched to in the profile P.

Note that the Smith-Waterman algorithm implicitly aligns two sequences in order to compute their similarity. We can make this alignment explicit: after we compute $F_{TP}(i, j)$, the best-matching position of the last node in T, N_{n_T}, is given by $J = \text{argmax}_j F_{TP}(n_T, j)$.

If the matched cluster contains the set of expanded trajectories $\{T_1 \dots T_k\}$, all of which have been aligned to be

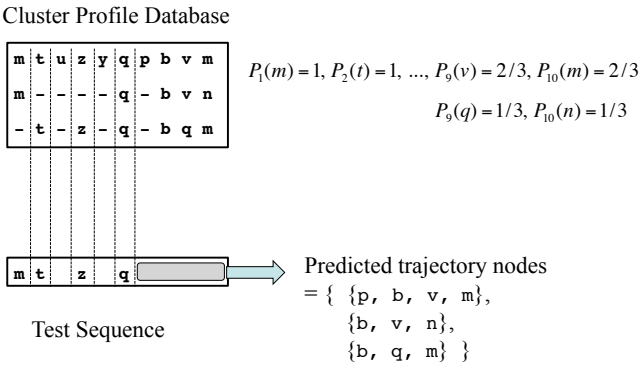


Figure 3: Sequence alignment of a partial trajectory with a cluster profile.

of length n as described in Sec. 4.3.1, then the set of trajectories that needs to be considered by the data stashing optimization is $\{T_1[J,n] \dots T_k[J,n]\}$. See Fig. 3 for an illustration.

6. OPTIMIZATION

Contrary to traditional routing schemes, data delivery by stashing does not route to the current position or in fact, to any single future position of a mobile node. Instead, we route to all possible trajectories of one or several mobile nodes. To this end, we choose a set of nodes that covers all trajectories, but at the same time is as cheap to route to as possible.

We formulate the problem of data delivery from a data source to stashing nodes along a set of trajectories as a linear programming relaxation of a binary integer program. The proposed scheme finds, for each data source, the optimal stashing nodes to which to send the data. Each sensor node can compute the solution to its particular routing problem independent of the other nodes. In the following, we will assume that a node A is asked to route data to one or several mobile nodes which travel along a set of possible trajectories $\{T_1 \dots T_m\}$. The output of the optimization is a set of stashing nodes $R = \{R_1 \dots R_k\}$.

To set up our linear program, let us first define an indicator function $I(N)$ indicating whether our data source has chosen N to be part of its set of stashing nodes:

$$I(N) = \begin{cases} 1 & N \in R, \\ 0 & \text{otherwise.} \end{cases}$$

Based on this definition, we can write the objective function to minimize as

$$f = \sum_N I(N) C(A, N), \tag{2}$$

where $C(\cdot, \cdot)$ denotes the routing cost between two nodes. In our experiments, we use the expected number of transmissions on a link as the routing cost for that hop, and the cost for a path is the sum of the per-hop costs.

In order to make sure that the data can be retrieved by the mobile sinks, there must be at least one stashing node on each of the trajectories. Given the trajectories $T_i = B_1^i \dots B_{n_i}^i$, we can write this condition as a single linear constraint per trajectory T_i:

$$\sum_{0 < j \le n_i} I(B_j^i) \ge 1 \tag{3}$$

Using these definitions, our problem is to find a set R that minimizes (2) subject to the constraints (3). This problem can be solved by a linear program (LP) if we ignore the integrality constraints. In our case, since the variable $I(N)$ is either zero or one, we are dealing with the special case of binary integer programming, which we solve using the bintprog optimization toolbox in MATLAB and AMPL/Gurobi.

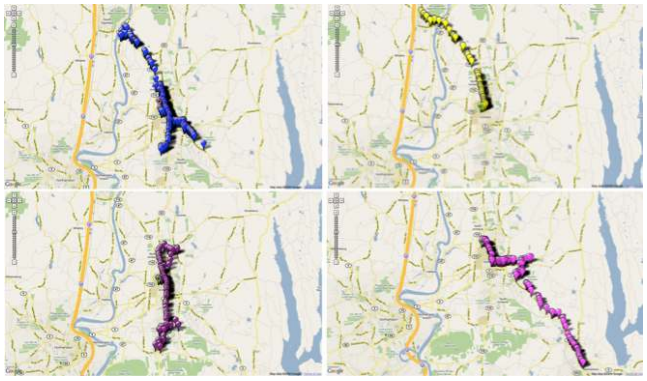


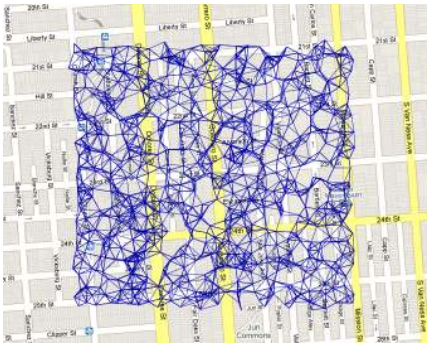
Figure 4: Typical trajectories of moving buses in UMass from the DieselNet dataset. When a bus is associated with a nearby access point, the access point is shown with a marker.

7. EVALUATION

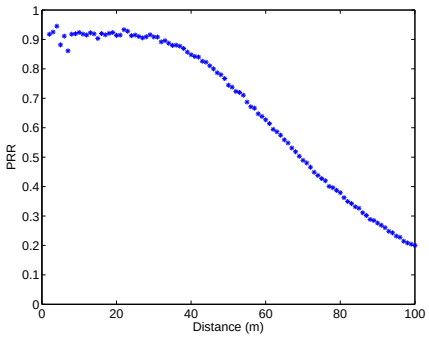
First we validate our trajectory clustering algorithm using real-world mobility data traces from UMass DieselNet [4] (shown in Fig. 4). The traces consist of time series of wireless access point (AP) IDs that wireless cards installed in buses connect to. There are 34 buses, 4198 access points, and 789 bus trips in the dataset, covering an area in and around the UMass campus.

We also test our algorithms in a simulated network deployed in downtown San Francisco. The network consists of 716 sensor nodes in an 830×790 m^2 area (see Fig. 5). We generated 20 different trajectories, a subset of which we show in Fig. 6. Each vehicle moves at a random speed of $\mathcal{N}(30, 5^2)$ km/h and broadcasts beacons at 1 Hz. To derive radio signal strengths for transmitted packets, we use a combined path-loss and shadowing model with a path-loss exponent of 3, a reference loss of 46.67 dB, and an additive Gaussian noise of $\mathcal{N}(0, 5^2)$ in dB. These parameters have been derived from measurements in urban environments [13]. We model interference effects using the CPM model [22] in TinyOS 2.1 [1] with `meyer-light` noise traces.

We implemented our routing algorithm in the TinyOS TOSSIM simulator [23] using idealized static shortest-path routing. In our scenario, it is often the case that we route several packets along similar paths. We use multicast to reduce redundant packet transmissions. We ran all of the experiments 10 times, and draw mean values with standard deviation error bars wherever applicable.



(a) Connectivity graph over 716 sensor nodes where links are shown for PRR > 75%.



(b) Wireless connectivity characteristic in simulation.

Figure 5: Wireless mesh sensor networks in downtown San Francisco for simulation. 716 **sensor nodes are distributed over** 830×790 **m^2.**

We evaluate routing in terms of routing cost, packet delivery, and load balance metrics, and compare our optimization scheme (*Stash*) to two other protocols: a point-to-point proactive distance-vector routing protocol (*Direct*) where each sensor node delivers its data to the currently connected relay node of each mobile sink, and the idealized stashing scheme that is given the perfect set of future locations for all sinks (*Stash(opt)*). The *Direct* protocol compares performance of our optimization scheme to traditional data delivery methods. The *Stash(opt)* scheme serves as an upper bound on what our algorithm could achieve, given perfect prediction. Note that this is not only a theoretical bound; it is achieved if the trajectories of nodes are known in advance — for example because the mobile sink announces them.

Our evaluation shows that *Stash* has lower control overhead than *Direct*. Both *Stash* and *Direct* require flooding that reaches the entire network to announce the presence and paths to the mobile sink. However, there is a key difference: *Direct* scheme requires continuous flooding to announce each mobile sink's current relays, while in the *Stash* scheme, the mobile sinks need to announce the anticipated trajectory node IDs only once. In our 716 node topology, it took 682 packet transmission to disseminate one packet from a mobile user to the entire network using Drip dissemination algorithm in TinyOS 2.x. In our simulation setting, the *Direct* method requires one position update every 2 seconds for the sink speed of 30 km/h. This position update needs to be disseminated throughout the network. Hence, the control overhead of *Direct* for this setting is 341 packet transmission per second. On the other hand, in *Stash*, the encoded set of trajectory nodes can be disseminated throughout the network with a total of 7502 packet transmissions per mobile sink. [1] Thus, the control overhead of *Direct* exceeds that of *Stash* after 22 seconds of operation and continuously increases at 341 packet transmissions per second while the overhead for *Direct* remains constant.

When we evaluate the routing cost, we count how many packets were used to deliver data from sensor nodes to destination nodes, after sensors learn the identity of the correct relay or possible relay candidates. In the evaluation below, we demonstrate that even without considering the control cost, our *Stash* scheme requires far fewer data packets than the *Direct* scheme.

In all of our experiments, we measure whether packets arrive at the stashing node (or in the direct routing case, at the current relay node), we do not take into account packet loss on the last hop, from the stashing or relay node to the mobile node. Since this affects *Stash* and *Direct* equally, it does not change the comparative analysis, however, it might lower the overall reliability of both methods. Consequently, we only count a packet as delivered if it is stashed at a node that is visited by the mobile node, i. e., if the stashing node is the best-connected sensor node to the mobile node at any point in time. In reality, even if the stashing node is never selected as the best-connected node, it might still be within range. While this would slightly increase the reliability of data stashing, we do not believe it would change the qualitative results.

Note that the protocols use global knowledge of the network and deliver data to mobile sinks along shortest routes. A specialized protocol like S4 [30] might be a better choice for the dynamic routing environment in sensor networks. To understand the implications of using scalable routing protocol such as S4 to route packets to the stashing nodes, we ran the S4 protocol in TOSSIM on the same topology with 20 beacon nodes in which we ran *Stash*. We computed the cost of the paths selected by S4 to route packets from the sensor nodes to the stashing nodes. The result shows that the routing cost of *Stash* using S4 is 1.27 times higher than if using an ideal shortest path routing. We do not expect this change in routing algorithm to lead to significantly different results of our comparative evaluation.

We demonstrate that given even limited information about future trajectories of sinks, optimization of routing paths leads to significant improvements in routing performance.

7.1 Clustering and Trajectory Prediction

We tested the hierarchical clustering algorithm described in Sec. 4 on the DieselNet dataset. The algorithm clustered the set of 789 bus trips into 23 clusters. Even though we have no ground truth to compare these clusters against, we visually evaluated the clusters and found them of good quality.

To make sure that our prediction would work in real-world settings, we use the clusters we found in the DieselNet traces to predict likely trajectories for a partial trajectory (which was not part of the training data). Since there is no net-

[1]The size of the encoded trajectory requires 11 packets due to 110 byte payload limit in TinyOS packets. Thus, it takes $7502 (= 682 \times 11)$ packet transmissions per mobile sink.

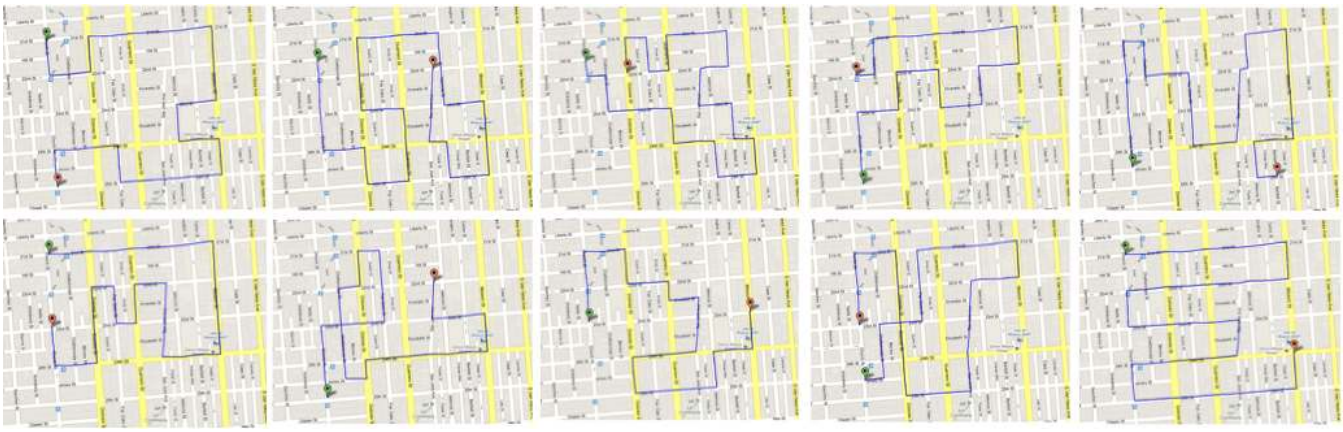


Figure 6: Moving paths of mobile vehicles. Each vehicle moves at a speed of $\mathcal{N}(30, 5^2)$ in km/h. We generate 20 different moving paths including the opposite direction as well. All of 20 vehicles are moving over the networks while communicating with sensor nodes as in Fig. 5(a).

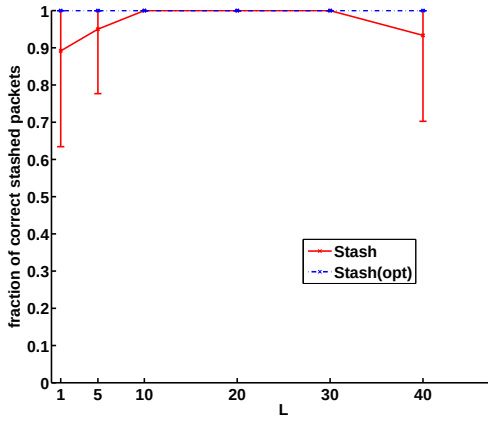


Figure 7: Fraction of packets stashed on nodes that are actually visited by the mobile node depending on number of nodes L used for prediction in the DieselNet dataset.

work data available, we assume that nodes are connected by perfect links, and that routing cost between two nodes is proportional to the Euclidean distance between them. While these idealized assumptions do not allow us to draw conclusions about network-related quality metrics, they help us evaluate the quality of our prediction algorithm in the context of data stashing. Using the predicted trajectories, and the cost metric described above, we select stashing nodes for ten randomly chosen data sources in the network, and measure what percentage of packets the mobile sink is able to retrieve. The results in Fig. 7 show that our prediction method results in excellent stashing node selections for real-world data.

7.2 Network Performance

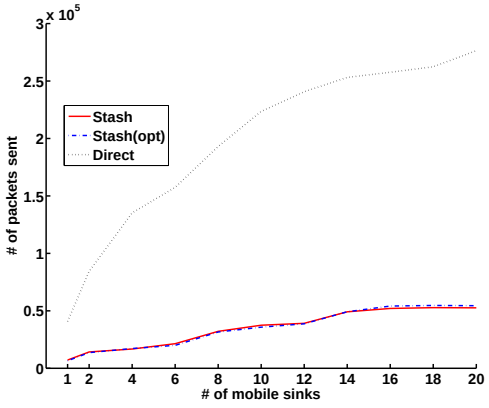
We evaluated our network optimization scheme against the direct point-to-point and perfect stashing algorithms using the simulated network. In these experiments, all 716 sensor nodes are transmitting data to 1 – 20 mobile sinks. Given the moving paths of mobile vehicles as shown in Fig. 6, we constructed trajectory clusters and their profiles. The average length of a cluster profile is 513.

We first analyze how the number of mobile sinks affects the performance of these algorithms. Even though the performance of all algorithms degrades as the number of sinks increases, stashing algorithms are affected less, because they exploit overlaps in the different trajectories (see Fig. 8). This effectively prevents network congestion. In fact, data stashing requires only 19% of packets to deliver the same data, compared to direct routing. Consequently, congestion in the network causes direct routing to drop a significant number of packets while stashing algorithms deliver above 80% of the packets even for 20 sinks (see Fig. 8(b)). The *Stash* routing algorithm uses up to 30 retransmissions just like the state-of-the-art collection protocol CTP [12]. Note that the performance of stashing algorithms also decreases due to increased network congestion, but at a much lower pace.

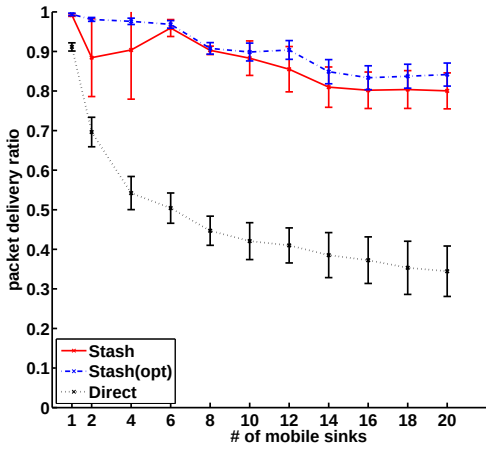
The performance of the predictive stashing scheme is close to the upper bound set by perfect prediction, suggesting that even limited knowledge of the future trajectory can significantly improve routing performance.

We also evaluate how the length of predicted trajectories affects performance. If the trajectory prediction is very uncertain far in the future, or if there are some constraints on permissible packet delivery delay, it might be preferable not to use the full predicted trajectories, but only allow stashing at the first W nodes. The results of these experiments are summarized in Fig. 9. Intuitively, longer trajectories give the network optimization more choice to select future stashing nodes. Consequently, sensors are more likely to find stashing nodes close to their own location, decreasing routing cost and congestion. Note that our optimization scheme can only counterbalance the effects of imperfect trajectory prediction if it is given enough choice. In our experiments, the break-even point is at $W = 10$. Achieving high reliability and efficiency of data delivery to the sinks, however, has its cost in increased delay. As W increases, it is more likely that the stashing nodes are located far in the future along the sink's trajectory.

There is another interesting tradeoff between transmission cost and computation cost depending on W. As W

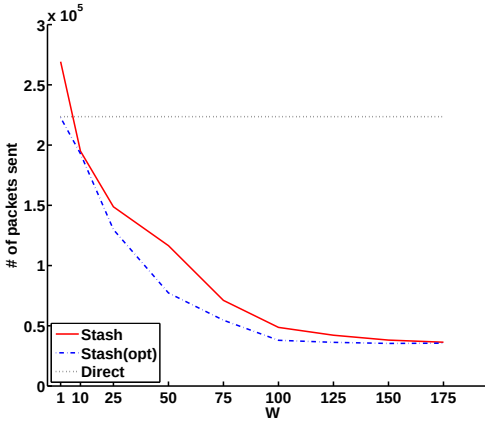


(a) Routing cost

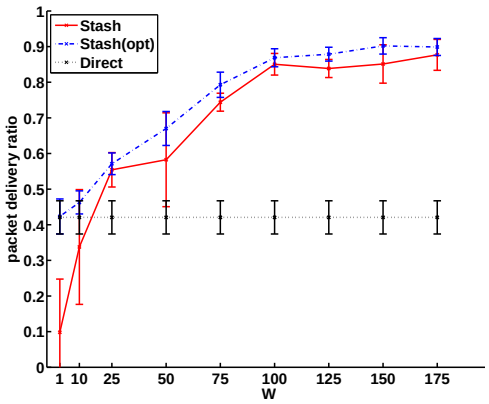


(b) Reliability. Shown are mean, error bars are standard deviation

Figure 8: Routing cost and delivery reliability depending on the number of mobile sinks.



(a) Routing cost



(b) Packet delivery ratio to mobile sinks, representing the mean value and error bars of standard deviation

Figure 9: Routing cost and delivery reliability depending on the number of predicted trajectory nodes W for 10 mobile sinks.

increases, each sensor node receives a larger number of anticipated trajectory nodes from mobile sinks, and needs to solve a more complex linear program. In practice, especially in large networks where we would expect very long trajectories, one would set a limit of $W \approx 100$.

The prediction algorithm uses the first L nodes of the sink trajectory to predict the rest. Fig. 10 shows the performance of data stashing (we use packet reception ratio as a proxy) as a function of L. Too little information about the trajectory leads to worse performance as prediction quality suffers. However, waiting for more information is only useful up to a point: waiting for information also results in fewer choices for stashing, since some of the trajectory has already been visited. In our setting, $L = 20$ appears optimal.

To evaluate the feasibility of efficiently computing the stashing nodes through optimization on the sensor node platform, we measured the running time for solving the binary integer program described in Sec. 6. The results for different platforms are shown in Fig. 11: we tested the performance on a Dell Precision 390 PC with Ubuntu Linux and a 2.4 GHz Core 2 Duo processor, and an embedded platform: a fit-PC2 with Ubuntu Linux and Intel Atom Z530 1.6GHz. We also tested two solvers: the bintprog optimization toolbox in MATLAB and the AMPL/Gurobi solver. The solution time for the optimization problem each node has to solve is less than 500 ms on an embedded platform.

A strength of data stashing is implicit load-balancing. Fig. 12 shows that data stashing spreads packet transmissions more evenly, as opposed to the tree-like routing patterns seen in direct routing to the current position of the mobile sink.

We have also tested the robustness of our data stashing scheme against differences in the speed of mobile users. Because the trajectory matching algorithm implicitly compensates for speed differences, changes in the speed of mobile users do not affect reliability. After training with a speed of 30 km/h, varying the speed between 30 and 90 km/h in the testing phase has no significant impact on reliability, which remains above 80% for $30, 50$ km/h and above 70% above for $70, 90$ km/h as shown in Fig. 13.

Finally, we evaluate the storage requirements that data stashing algorithms impose on sensor nodes (see Fig. 14). It is likely that data stashing requires more storage than direct routing schemes; the node stashing most data needs to store around 200 packets in our scenario. Such peaks occur at "favorite" stashing locations, such as the intersection of several trajectories. In our opinion, data storage is generally

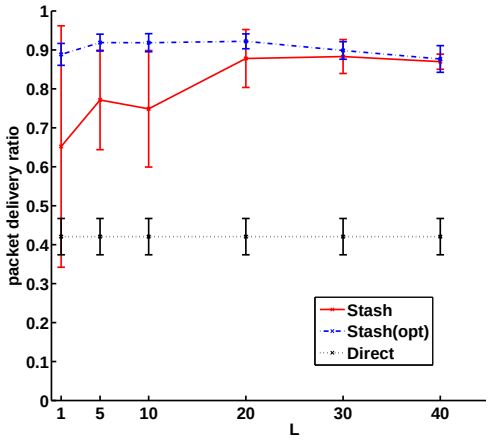


Figure 10: Packet delivery reliability depending on number of nodes L used for prediction. Shown is data for 10 mobile sinks, with mean value and error bars showing standard deviation.

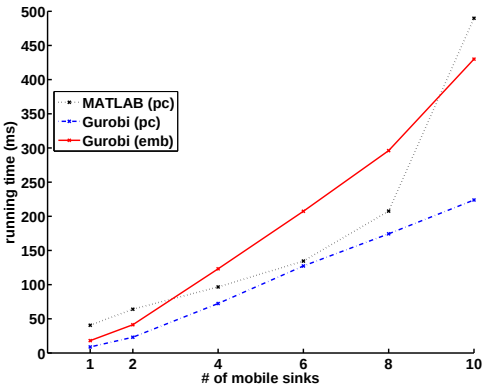


Figure 11: Running time for a sensor node to solve an optimization problem for stashing in each platform/tool depending on the number of mobile sinks.

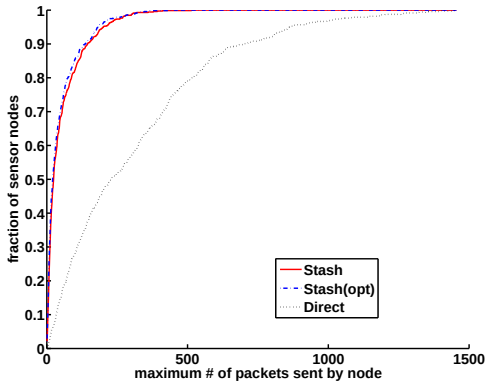


Figure 12: Fraction of nodes sending less than a certain number of packets (for 10 mobile sinks case).

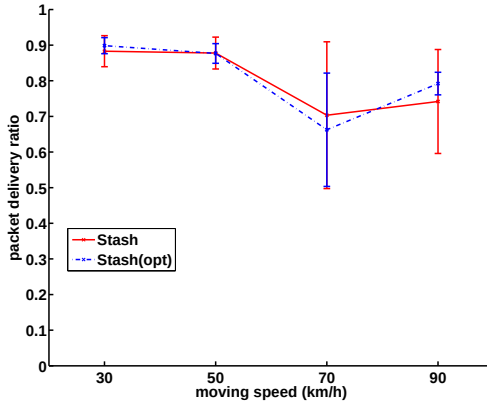


Figure 13: Packet delivery reliability depending on speed of mobile users. Shown is data for 10 mobile sinks, mean value and error bars showing standard deviation.

less problematic than radio transmission in sensor networks, making this a good trade-off.

8. CONCLUSION

Energy concerns are extraordinarily important for practical deployments of sensor networks. Radio transmission consumes a large part of the limited energy resources of sensor nodes. We have presented a data delivery protocol that exploits knowledge of the mobility of sinks querying the sensor network to reduce transmission cost. We focused on the common case that the sensor data is delay-tolerant. Instead of directly transmitting to the mobile sink, data can be stashed along the sink's trajectory, where it will be picked up when the mobile sink passes.

Our experiments indicate that our scheme significantly decreases the total transmission cost for providing the requested information to mobile sinks. We also show that we can provide much better load-balancing, avoiding collisions and consuming energy resources evenly throughout the network, leading to longer overall network lifetime. More importantly, we demonstrate that given limited information about future trajectories of sinks, optimization of routing paths leads to significant improvements in routing performance. Our proposed method provides not only a mobile routing protocol, but rather a way to improve any existing protocol by learning and exploiting mobility patterns.

Currently, we only select stashing nodes once and do not monitor the progress of the mobile sinks as they move through the network. In scenarios where prediction is more difficult, recomputing the set of stashing nodes and correcting prediction errors by re-stashing at newly predicted nodes could significantly increase robustness.

Although our method can take into account multiple mobile sinks without problems, there is currently no protocol that accounts for the possibility of announcing several sinks at once. This straightforward extension of our method would be useful in several scenarios.

The trajectory clustering algorithm is currently executed in an off-line learning phase. However, our proposed scheme does not necessarily require a separate off-line phase. As each mobile device keeps updating its own trajectory model, each mobile node can predict its own anticipated trajectory using a local model. If the network size is very large, it may not be feasible to maintain huge databases of mobility trajectories in a mobile device. In the future, we anticipate

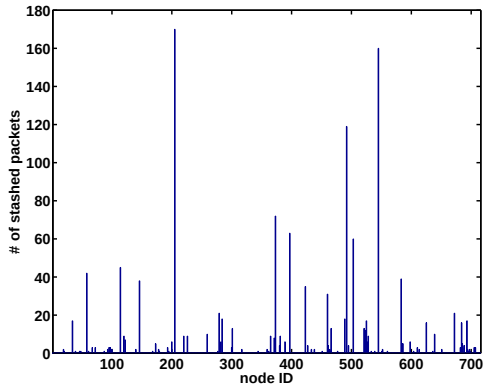


Figure 14: Storage overhead over the sensor nodes for 10 mobile sinks.

working on distributed or hierarchical computation and storage of the mobility models.

Interesting directions for algorithmic improvements include a more sophisticated clustering method that explicitly represents partial trajectories and is able to partition long trajectories into short pieces that can be clustered more efficiently. A multi-tier or hierarchical approach to deal with extremely large networks is another avenue for future work.

9. ACKNOWLEDGMENTS

The authors would like to thank Prof. Serafim Batzoglou and Chuong (Tom) Do for useful discussions and comments. We gratefully acknowledge the support of NSF grants ITR 0205671, CNS 0619926, CNS 0626151, and ARO grants W911NF-06-1-0275, W911NF-07-2-0027, as well as a fellowship from the Samsung Scholarship.

10. REFERENCES

[1] TinyOS 2.1.0. http://www.tinyos.net/tinyos-2.1.0/.

[2] A. Agrawal and S. K. Khaitan. A new heuristic for multiple sequence alignment. In *Proceedings of the IEEE International Conference Electro/Information Technology*, 2008.

[3] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, Jan 2003.

[4] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine. Relays, base stations, and meshes: enhancing mobile networks with infrastructure. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 81–91, New York, NY, USA, 2008. ACM.

[5] M. Bayir, M. Demirbas, and N. Eagle. Mobility profiler: A framework for discovering mobile user profiles (technical report version). *cse.buffalo.edu*, 2008.

[6] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *IPSN '03: Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, Palo Alto, CA, USA, 2003.

[7] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(4):609–619, 2004.

[8] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM.

[9] J. Froehlich and J. Krumm. Route prediction from trip observations. *SAE SP*, Jan 2008.

[10] S. Gandham, M. Dawande, R. Prakash, and Subbarayan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *GlobeCom '03: Proceedings of the Global Communications Conference*, San Francisco, CA, USA, 2003.

[11] J. Ghosh, M. Beal, H. Ngo, and C. Qiao. On profiling mobility and predicting locations of campus-wide wireless network users. *Technical Report: State University of New York at Buffalo*, Jan 2005.

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, November 2009.

[13] A. Goldsmith. *Wireless Communications*. Cambridge University Press, New York, NY, USA, 2005.

[14] D. Johnson, D. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad Hoc Networking*, 2001.

[15] S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, September 1967.

[16] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 193–204, New York, NY, USA, 2003. ACM.

[17] D. Kotz, T. Henderson, and I. Abyzov. CRAWDAD data set dartmouth/campus (v. 2004-12-18). Downloaded from http://www.crawdad.org/dartmouth/campus, Dec. 2004.

[18] J. Krumm. Real time destination prediction based on efficient routes. *Society of Automotive Engineers (SAE) 2006 World Congress*, Jan 2006.

[19] B. Kusy, H. Lee, M. Wicke, N. Milosavljevic, and L. Guibas. Predictive qos routing to mobile sinks in wireless sensor networks. In *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 109–120, Washington, DC, USA, 2009. IEEE Computer Society.

[20] K. Laasonen. Clustering and prediction of mobile user routes from cellular data. *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 2005.

[21] K. Laasonen, M. Raento, and H. Toivonen. Adaptive on-device location recognition. *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 2004.

[22] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th international conference on*

Information processing in sensor networks, pages 21–30, New York, NY, USA, 2007. ACM Press.

[23] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Simulating large wireless sensor networks of tinyos motes. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[24] Y. Li, J. Harms, and R. Holte. Optimal traffic-oblivious energy-aware routing for multihop wireless networks. In *INFOCOM '06: Proceedings of the 26th Conference on Computer Communications*, Barcelona, Spain, 2006.

[25] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, Jan 2007.

[26] L. Liao, D. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, Jan 2007.

[27] L. Lin, N. B. Shroff, and R. Srikant. Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources. *IEEE/ACM Trans. Netw.*, 15(5):1021–1034, 2007.

[28] J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM '05: Proceedings of the 25th Conference on Computer Communications*, Miami, FL, USA, 2005.

[29] J. Luo, J. Panchard, M. Piorkowski, M. Grossglauser, and J.-P. Hubaux. Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In *DCOSS '06: Proceedings of the International Conference on Distributed Computing in Sensor Systems*, San Francisco, CA, USA, 2006.

[30] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *4th Symposium on Networked Systems Design and Implementation (NSDI 2007)*, 2007.

[31] C. Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, January 2002.

[32] P. Nurmi and J. Koolwaaij. Identifying meaningful locations. *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*, pages 1 – 8, Jul 2006.

[33] 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).

[34] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, pages 30–41, 2003.

[35] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[36] L. Song, U. Deshpande, U. Kozat, D. Kotz, and R. Jain. Predictability of wlan mobility and its effects on bandwidth provisioning. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 – 13, Apr 2006.

[37] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive wi-fi mobility data. *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2:1414 – 1424 vol.2, Feb 2004.

[38] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–4680, November 1994.

[39] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[40] R. Wohlers, N. Trigoni, R. Zhang, and S. Ellwood. Twinroute: Energy-efficient data collection in fixed sensor networks with mobile sinks. In *MDM '09: Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 192–201, Washington, DC, USA, 2009. IEEE Computer Society.

[41] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, New York, NY, USA, 2002. ACM.

[42] J. Yin, Q. Yang, D. Shen, and Z.-N. Li. Activity recognition via user-trace segmentation. *Transactions on Sensor Networks (TOSN)*, 4(4), Aug 2008.