Predictive Data Delivery to Mobile Users Through Mobility Learning in Wireless Sensor Networks

HyungJune Lee, *Member, IEEE*, Martin Wicke, Branislav Kusy, *Member, IEEE*, Omprakash Gnawali, *Member, IEEE*, and Leonidas Guibas, *Fellow, IEEE*

***Abstract*—We consider applications, such as indoor navigation, evacuation, or targeted advertising, where mobile users equipped with a smartphone-class device require access to sensor network data measured in their proximity. Specifically, we focus on efficient communication protocols between static sensors and users with changing location. Our main contribution is to predict a set of possible future paths for each user and store data at sensor nodes with which the user is likely to associate. We use historical data of radio connectivity between users and static sensor nodes to predict the future user-node associations and propose a network optimization process, i.e., *data stashing*, which uses the predictions to minimize network and energy overheads of packet transmissions. We show that data stashing significantly decreases routing cost for delivering data from stationary sensor nodes to multiple mobile users compared with routing protocols where sensor nodes immediately deliver data to the last known association nodes of mobile users. We also show that the scheme provides better load balancing, avoiding collisions and consuming energy resources evenly throughout the network, leading to longer overall network lifetime. Finally, we demonstrate that even limited knowledge of the location of future users can lead to significant improvements in routing performance.**

***Index Terms*—Data delivery to mobile users, mobility pattern, network optimization, sensor networks, trajectory prediction.**

I. Introduction

CLASSIC multihop wireless routing protocols compute the shortest path in terms of hops or expected number of transmissions (ETX) [68] between sources and destinations in a network. Energy consumption for radio transmissions corresponds to a considerable portion of the total energy consumption at sensor nodes [17]. Since the shortest path minimizes the number of necessary transmissions, this strategy minimizes not only delay but energy used for data communication as well.

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 sensor nodes (as *data sources*) 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). Data sinks (humans or vehicles) move inside the area covered by the sensor network and access sensor data through computationally capable devices, such as smartphones. 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 [16]. 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. For instance, location-sensitive data such as store advertisement and customized evacuation notifications [21] can be forwarded to specific mobile users as they come around in the network.

There is a large body of prior work in the field of routing from nodes to mobile sinks. We can classify them into two categories: 1) proactive scheme, such as optimized link state routing (OLSR) [13] and destination-sequenced distance vector routing [47]; and 2) reactive scheme, such as dynamic source routing (DSR) [24] and ad hoc on-demand distance vector routing (AODV) [46]. The state-of-the-art ad-hoc routing protocols can discover routes without initially knowing the topology of the networks, and this aspect is considered a big advantage of these protocols over traditional routing protocols such as Open Shortest Path First [14] and Routing Information Protocol [39]. However, the problem is that their routing performance degrades rapidly with increasing mobility, i.e., resulting in higher route update cost for proactive schemes or higher bandwidth usage of on-demand flooding for reactive schemes, as investigated in [53].

To design a robust routing algorithm under sinks' mobility, exploiting some trajectory information of the mobile nodes may be necessary. In some applications, sinks know their

Manuscript received February 24, 2014; revised July 29, 2014 and October 27, 2014; accepted December 5, 2014. Date of publication January 1, 2015; date of current version December 14, 2015. This work was supported by the National Research Foundation of Korea funded by the Korean Ministry of Science, ICT, and Future Planning through the Basic Science Research Program under Grant NRF-2013R1A1A1009854. The review of this paper was coordinated by Dr. T. Taleb.

H. Lee is with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 120-750, Korea (e-mail: hyungjune.lee@ewha.ac.kr).

M. Wicke is with the eddy.systems, San Francisco, CA 94102 USA (e-mail: martin.wicke@gmail.com).

B. Kusy is with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Pullenvale, QLD 4069, Australia (e-mail: branislav.kusy@gmail.com).

O. Gnawali is with the Department of Computer Science, University of Houston, Houston, TX 77004 USA (e-mail: gnawali@cs.uh.edu).

L. Guibas is with the Department of Computer Science, Stanford University, Stanford, CA 94305 USA (e-mail: guibas@cs.stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2014.2388237

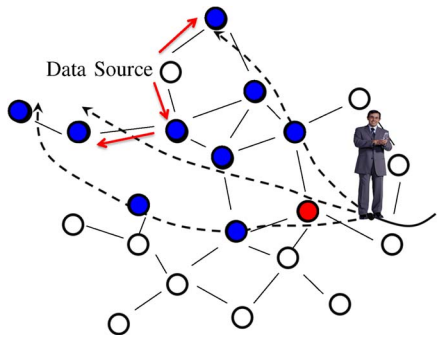


Fig. 1. Overview of our routing algorithm using the anticipated association nodes. The red node is the predicted next association node (through *short-term* prediction [28]), and the blue nodes are sequences of the future association nodes (through *long-term* prediction). Packets can be *stashed* for pickup at blue nodes.

future trajectory through the network and can announce it to the network when requesting information. Even if the future trajectory is unknown, since many applications are deployed in environments that constrain motion patterns of sinks to roads, trails, or hallways, not all possible movements within space are actually realized. Moreover, a recent study [56] has showed that mobile users move along a limited set of typical spatial trajectories, and the movement shows a certain degree of regularity. This suggests that we can learn the structure in users' movements, which is called the *mobility pattern* from repeated observations, and exploit the mobility pattern for designing a more reliable and efficient routing scheme that works even under high mobility.

We present a *long-term mobility prediction* algorithm that allows us to predict a sequence of node associations of mobile sinks, as shown in Fig. 1. To do this, we present a method for representing trajectories with wireless association, learning typical trajectories from observations, and predicting likely association patterns given observed partial association history. We borrowed ideas of sequence similarity, clustering, and alignment from computational biology. Wireless devices carried by mobile sinks run the prediction algorithm to compute and supply information about their future association sequences to the network. We define *trajectory* as a sequence of node associations and compute similarities between sequences in the association data acquired in a learning phase. Using these similarities, we compute clusters representing typical paths through the network. We then compute a compact probabilistic representation for the clusters that we can use to efficiently find likely future trajectories during prediction.

Based on the long-term mobility prediction algorithm described earlier, we design a routing scheme that exploits knowledge about the long-term association pattern of mobile sinks within a network of data sources. It aims to minimize energy consumption and network congestion. This enables the routing scheme to scale to multiple mobile sinks and a large number of data sources. For delay-tolerant network applications, which do not require immediate real-time data retrieval, we propose to route data not to the mobile sink directly, but to relay nodes along a predicted path of the mobile node that is close to the data source in terms of communication hops (see Fig. 1). The selected relay node will *stash* the information to be picked up when the mobile node passes within the transmission range of the relay node. We use an integer programming technique to find optimal relay nodes that minimize the number of necessary transmissions while guaranteeing robustness against link and node failures and achieving better load balancing and more even utilization of network resources.

Our main contributions can be summarized as follows.

- We present *data stashing*, which is a data delivery scheme that routes data to mobile sinks. In this scheme, each sensor node selects a set of nodes on which its data will be stashed, so that the overall network and energy costs of delivering data to one or more users are minimized.
- 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 to efficiently find the best-matching cluster given a partial trajectory.

This paper extends our prior work presented in [32] as follows.

- We present a new optimization formulation that allows our algorithm to trade off routing efficiency for improved data latency.
- We validate our proposed *data stashing* scheme in a real-world test bed where 41 TelosB sensor nodes are deployed, showing practical applicability.
- We perform qualitative and quantitative analyses on the effects of packet delivery delay and energy saving in terms of network routing performance in a large-scale simulated network.
- We add experimental results to show the geographical load balancing in packet transmission over the network.
- We add experimental results to investigate the geographical location of the selected stashing nodes over the network.
- We add discussions on the applicability of the technique in different types of networks, and the possibility of cross-layer integration with the media access control (MAC) layer.

The remainder of this paper is organized as follows. After discussing related work in Section II, we present our long-term mobility prediction algorithm in Section III. In Section IV, we propose our predictive data delivery schemes, and Section V presents the evaluation results of our proposed approach. After we add discussions in various aspects in Section VI, we conclude this paper in Section VII.

II. Related Work

Related work can be classified into two categories: mobility prediction and routing to mobile users.

A. Long-Term Mobility Prediction

Long-term mobility pattern modeling has been studied using GPS data or association data from cellular networks or wireless local area networks (WLANs). Because raw GPS data contain many outliers, most of the previous research [5], [27] filters out noisy and unreasonable measurements first. They then identify the possible goal locations from the filtered GPS positions and construct prediction models. Alvarez-Garcia *et al.* [5] found places where a user spent a significant amount of time and clustered them into locations. A hidden Markov model was applied to movement between locations, which is then used to predict future locations. Krumm *et al.* [27] obtained end-to-end routes from raw GPS data and used a Bayesian model and a trip similarity clustering algorithm to predict the next location. Further, Lane *et al.* [29], and Lee *et al.* [33] not only extracted significant places from filtered GPS data but also tried to infer human activity associated with each different place. In [29], a user-specific activity classification model was constructed by embedding interperson similarity in various aspects of similarity networks. Moreover, in [33], a Bayesian network model for activity using context hierarchy based on contextual information from a mobile phone was presented. Their work suggest exploiting a high-level context (i.e., users' activities) of a mobile user with higher fidelity.

These previous approaches, including [43], infer long-term destinations of mobile users. Recently, in [42], a path prediction model has been proposed based on historical movement trace maps. This paper provides not only the destination prediction but can also predict all possible future *trajectories* of the user. Furthermore, our techniques rely only on wireless association traces, allowing more generic applicability of mobile users' movement.

Similarly, in cellular networks, some previous work [7], [23] predicts the next cell connection based on various information such as past handoff rate, size of the active set, active set update rate, and signal strength variation. In WLANs, a theoretical work [3] constructs a Gauss–Markov mobility model for predicting the speed, direction, and degree of randomness of mobile users. Further, based on real-world empirical traces, a long-term large-scale measurement study of user access point (AP) association at Dartmouth [26] 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-two Markov predictor [58] for users with long trace lengths. Further analysis of the same data set has suggested the feasibility of predicting the future associations of a mobile user in space and time [57], and a similar study [60] has been conducted for seamless handoffs in WLANs. Using a different data set, Ghosh *et al.* [18] described techniques to predict a user's location with respect to social hubs such as buildings and classrooms, rather than individual wireless APs. Although the approaches work with real-world mobility data and use only association data for predicting the future association, they do not explicitly deal with noisy association for classifying mobility pattern clusters. This paper offers an explicit mechanism for classifying user mobility patterns into different representative clusters, even if the wireless association trajectories are noisy.

More recent work has exploited user context information to distribute resources more efficiently across the network and as a basis for energy-efficient design of network applications [2], [21], [52]. Similarly, our work exploits users' mobility patterns from wireless traces, and our network optimization utilizes the predictive knowledge to improve energy efficiency of routing.

B. Routing to Mobile Users

There is a large body of research in routing protocols designed to deliver packets to mobile users in wireless networks. Some of these protocols assume little about the network and the mobility pattern of the mobile users and perform network discovery proactively or on demand. Classic protocols such as DSR [24] and AODV [46], which were originally designed for wireless ad hoc networks, and sometimes used in mobile ad hoc routing, fall into this category. In the wireless sensor network context, protocols such as directed diffusion [22], scalable energy-efficient asynchronous dissemination protocol [25], and two-tier data dissemination [65] construct energy-efficient routing paths without knowledge of the mobility patterns of the sink.

In particular, regarding routing to mobile sinks, mobile trajectories, or their sojourn times can be programmed to optimize data forwarding efficiency in [4], [38], [54], and [69]. This paper does not assume a programmable trajectory of the mobile sinks. Researchers have formulated computing energy-efficient routes in sensor networks as an optimization problem in [45], [62], and [63]. This paper also frames routing as an optimization problem. However, in our optimization 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.

There has been previous work on exploiting predicted mobility to improve the efficiency of routing to mobile users with predictable trajectories. Chatzigiannakis *et al.* [11] demonstrated an important claim that data collection should be adapted depending on characteristic mobility patterns for reducing energy consumption in routing. Chakrabarti *et al.* [10] proposed a protocol in which the mesh nodes keep statistics of user visits and transmit information only when the mobile user is within transmission range. This paper does not assume that the trajectory of a mobile node takes it within a single-hop transmission range of each mesh node in the network. Most closely related to this paper is the recent work on the proactive scheme in [61] and [67]. Based on the user arrival statistics, a subset of nodes elect themselves as storage nodes and initiate routing tree construction as roots. The mesh network forwards data to these storage nodes so that packets can be relayed to the mobile user. Although our work fits in this general framework, we employ different methods to overcome shortcomings of this approach. Our protocol is based on a clustering algorithm to improve the

accuracy of trajectory prediction (as described in Section III) and uses the predictive knowledge to help an efficient routing decision, which is scalable for many mobile sinks in terms of radio energy consumption and packet delivery reliability.

III. Long-Term Mobility Prediction

We predict likely long-term association nodes of mobile sinks by using the current association and a past history of association trajectories. We present a method for learning typical movement patterns from observations, representing trajectories and predicting likely trajectories, given observed partial trajectories. The prediction algorithm is used by the mobile node to compute and supply information about its future trajectory to the network. We characterize the trajectories as sequences of node associations and find clusters representing typical trajectories. Using multiple sequence alignment techniques to identify similar and dissimilar regions within a cluster, we then compute a compact probabilistic representation for the clusters that we use to efficiently determine likely future trajectories during prediction.

The predicted long-term trajectory of mobile sinks can benefit network applications; this is particularly obvious in efficient data delivery to mobile sinks. For example, when an information source needs to deliver data to multiple mobile sinks, it can select intermediate storage nodes that are close to the source in terms of communication hops and lie along the anticipated trajectory of the sinks. *Stashing* the data at the selected nodes, instead of routing the data directly to the sinks at their current positions, allows a mobile sink passing through the network to collect the data at intermediate storage nodes. We can further reduce redundant packet transmissions by sharing data deliveries via intermediate storage relays on nodes contained in several sinks' trajectories. The long-term connectivity prediction thus enables a scalable data delivery scheme for multiple mobile sinks.

A. Constructing the Mobility Model

We introduce a *trajectory* in terms of wireless association and present our mobile trajectory clustering method using given trajectories for an offline learning phase.

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 with no environmental restrictions, there are usually few interesting start and end points for any given journey, and sinks often follow short(or the shortest) paths from a starting point to a destination, 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 offline learning phase. 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 describes these concepts in turn.

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 associated sensor node at any given time.

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

$$T = N_1 N_2 N_3, \ldots, N_k.$$

We only record changes in the associated node list, 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 path 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. Moreover, note that trajectory data can be collected anonymously; the corresponding mobile sink ID is not needed, which helps to mitigate possible privacy concerns.

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 [55] to calculate this similarity measure between two sequences.

Informally, to compute the similarity between two sequences $T_A = A_1, \ldots, A_{n_A}$ and $T_B = B_1, \ldots, B_{n_B}$, 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 lengths i and j of T_A and T_B, $A_1, \ldots, A_i$ and $B_1, \ldots, B_j$. F_{AB} can be defined recursively as follows:

$$F_{AB}(i, 0) = 0 \quad \text{for} \quad 0 \leq i \leq n_A \tag{1}$$

$$F_{AB}(0, j) = 0 \quad \text{for} \quad 0 \leq j \leq n_B \tag{2}$$

$$F_{AB}(i, j) = \max [F_{AB}(i-1, j-1) + s(A_i, B_j), F_{AB}(i-1, j) + d, F_{AB}(i, j-1) + d, 0] \tag{3}$$

where the similarity for insertion or deletion operations d and 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. We define a per-node similarity function $s(A_i, B_j)$ where we set $s(A, A) = 1$ and $s(A, B) = 0 \ \forall A \neq B$, meaning that we penalize for a different node in substitute operations. 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 defined earlier, $F_{AB}(n_A, n_B)$ 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 the length of A or B, we normalize the similarity measure by dividing by the length of the shorter sequence

$$\text{sim}(A, B) = \frac{F_{AB}(n_A, n_B)}{\min(n_A, n_B)}. \quad (4)$$

Note that the similarity measure we define is not a distance metric.

3) Cluster Representation: Based on the pairwise similarities between all pairs of sequences, we apply a hierarchical clustering method. We use the *average linkage* metric that 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, see [41].

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. Although 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.

a) Multiple sequence alignment: Given a set of sequences, multiple sequence alignment algorithms compute how the sequences should be lined up 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 [64]. Heuristic alignment methods are widely used for DNA or protein alignments in bioinformatics [35]. We use a modified version of ClustalW, which is one of the most popular alignment tools [59].

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 modify ClustalW to use the set of node IDs instead of an alphabet of amino acids or DNA base pairs. 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 [36]. 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.

b) 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 $P = P_1, \ldots, P_n$. 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 zeroth-order Markov model of the set of aligned sequences.

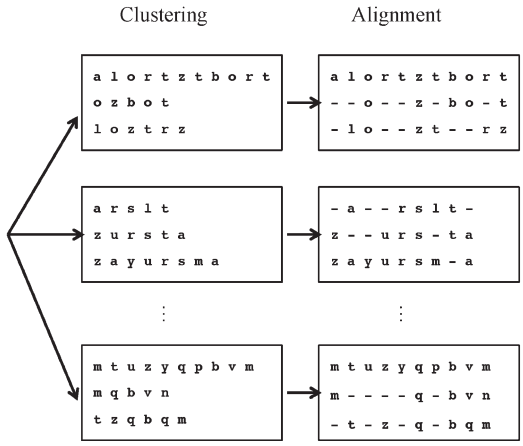


Fig. 2. Clustering and alignment procedures. A number of trajectories are clustered together with respect to sequence patterns and are aligned by using a multiple sequence alignment algorithm (ClustalW). The aligned sequences form a probabilistic trajectory profile.

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 of sequences after clustering. Sequences classified in a cluster [see Fig. 3(a)] are aligned to one another through the ClustalW algorithm as in Fig. 3(b). Given a group of these aligned sequences for each cluster, we calculate the probability distribution over each column index. In this way, we obtain a probabilistic trajectory profile for a cluster and continue this procedure for the other clusters.

For illustration purposes, we generate a graphical representation of probabilistic trajectory profiles [see Fig. 3(c)] by using WebLogo [15]. The figure graphically represents all possible realizations of sequences within a mobile trajectory cluster. The height of the letters within each stack indicates the relative frequencies for each possibility, whereas the width of the stacks indicates the relative proportion of valid readings at that position where the more gaps (i.e., spaces to compensate for insertions and deletions) in the sequence at a specific position means a thinner stack.

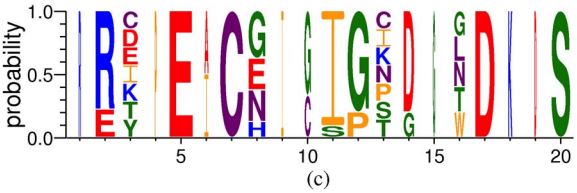
B. Connectivity Prediction

Here, we describe a long-term connectivity prediction algorithm. The prediction algorithm provides a set of possible future trajectory nodes using the mobile trajectory clusters constructed in Section III-A.

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

```
RTEACEGIPDS          -RT-EACE-GIP----D--S
RECEIGIPSDS          -R--E-CEIGIPS---D--S
YECI                 --Y-E-C---I---------
RECEICGIGNGNDS       REC-EICG--IGNG-ND--S
EDECIGPDS            -ED-E-C---IGP---D--S
RECHCIGKDS           -R--E-CH-CIGK---D--S
RECIGC               -R--E-C---IGC-------
RIECGSGDLDKS         -RI-E-CG--SG-D-LDK-S
KECGIGTDWDS          --K-E-CG--IGTD-WD--S
RECNIGDGTDS          -R--E-CN--IG-DGTD--S
REPECNIGIDGDKDS      -REPE-CN--IGID-GDKDS
```

(a) (b)



(c)

Fig. 3. Sequences belonging to a cluster, the aligned sequences, and their graphical profile generated by WebLogo [15]. (a) Sequences in a cluster. (b) Sequences after alignment. (c) Profile presentation from the aligned sequences in a cluster.

Specifically, we are given a partial trajectory $T_M = N_1, \ldots, N_{n_M}$ 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 pushing data to nodes that have already been visited by the mobile node. In the following, we describe how we compute the closest cluster (see Section III-B1) and how we compute the current position of the mobile node within the returned set of sequences (see Section III-B2).

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 (2) can be used unaltered, except that the partial match function F_{TP} now operates on a trajectory $T = N_1, \ldots, N_{n_T}$ and a profile $P = P_1, \ldots, P_{n_P}$. We need to change the definition of the per-node similarity function $s(N_i, P_j)$ (instead of using $s(A_i, B_j)$) to reflect the likelihood of N_i given the probability distribution P_j. We choose

$$s(N_i, P_j) = \begin{cases} eP_j(N_i), & P_j(N_i) > 0 \\ h, & \text{otherwise} \end{cases} \tag{5}$$

and use the parameter values $d = -1$ in (2), $e = 8$, and $h = -1$, which have been proven effective in our setting. By varying each parameter, we choose a set of parameters that leads to the most accurate cluster prediction for our datasets (see Section V). For instance, denser deployments incur higher variability of relay nodes; thus, the parameters need to allow for additional mismatches and insertions/deletion.

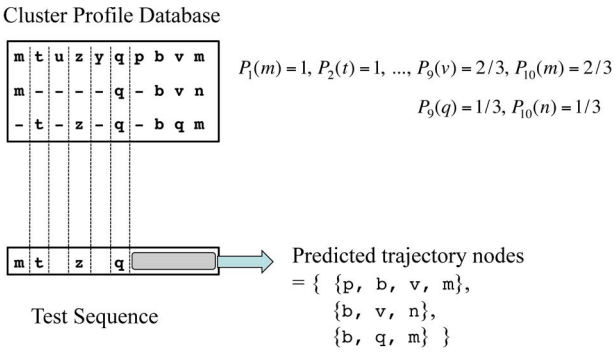


Fig. 4. Sequence alignment of a partial trajectory with a cluster profile.

2) Alignment: Once we have found the best-matching cluster, we need to align the partial trajectory with the sequences in the cluster 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 Section III-A3. 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 to which the last node in the partial trajectory N_{n_T} is matched in the profile P.

Note that the Smith–Waterman algorithm implicitly aligns two sequences 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 $J = \arg\max_j F_{TP}(n_T, j)$.

If the matched cluster contains the set of expanded trajectories $\{T_1, \ldots, T_k\}$, all of which have been aligned to be of length n, as described in Section III-A3, then the set of trajectories that needs to be considered by the data stashing optimization is $\{T_1[J, n], \ldots, T_k[J, n]\}$. See Fig. 4 for an illustration.

IV. Predictive Data Delivery

The main objective of this paper is to develop a routing scheme that delivers data to mobile sinks through a sensor network. We exploit knowledge about the mobility of the sinks to lower the cost and increase the reliability of data transmission.

In particular, we solve the following problem: One (or several) mobile sink moves through a network, collecting local data from the nodes in the network. Traditionally, we would either send all data directly to the current position of the mobile sink (i.e., 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 sink or if moving within the radio range of each desired node is not an option. We choose a compromise between the two extremes. Using knowledge about the trajectory of the mobile sink, data sources route data to a set of *stashing nodes* that store information along the likely trajectories of the mobile sink.

At the core of our method is an optimization procedure that chooses for each sensor node a set of stashing nodes

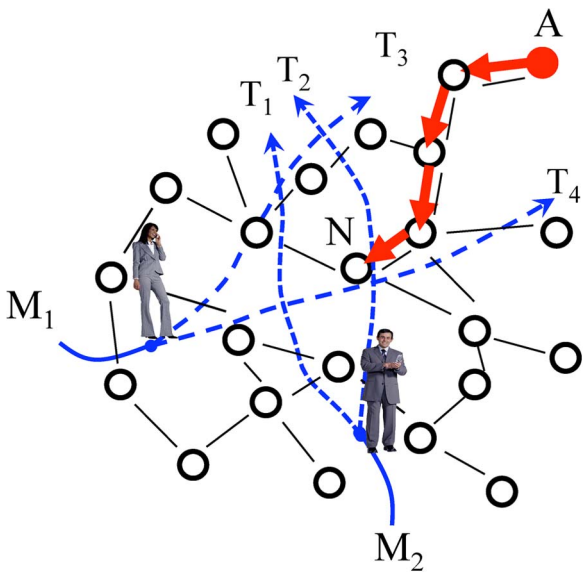


Fig. 5. Our optimization procedure chooses a set of nodes that covers all possible future trajectories (blue dashed line) of mobile sinks but, at the same time, is as cheap to route to as possible.

that guarantee (with high probability) that a mobile sink will receive data from the source (see Fig. 5 for an illustration). The optimization procedure is described in detail in Section IV-B.

We assume some knowledge about the possible trajectories that a mobile sink can take. This information is either provided by the mobile sink or is deduced from motion patterns of sinks in the network, as shown in Section III.

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

A. Protocol

Here, we clarify the overall procedure of trajectory prediction, stashing node selection, and routing. We provide a high-level description of the protocol used to negotiate data stashing for a mobile sink, as shown in Fig. 6. 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* for proxy. As the sink moves through the network, we can observe a sequence of relay node IDs. We use this sequence to predict a set of likely trajectories that most closely match the recorded prefix in the database of historical trajectories acquired in an offline learning phase, as described in Section III-B. 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 sink.
3) Stashing node selection. Upon receiving a sink's request for data and a set of likely trajectories, each sensor node (which is a data source) 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 Section IV-B.
4) Data stashing. Sensor nodes forward data to the stashing nodes for future delivery to mobile sinks.
5) Data collection. As the mobile sink 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 sink.

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 Section V.

Note that we assume an underlying point-to-point routing protocol such as S4 [40]; however, we make no assumptions on the properties of this protocol.

B. Network 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 data source 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 that travel along a set of possible trajectories $\{T_1, \ldots, T_m\}$. The output of the optimization is a set of stashing nodes $R = \{R_1, \ldots, R_k\}$.

To set up our integer 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.e.,

$$I(N) = \begin{cases} 1, & N \in R \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

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

$$f = \sum_{N} I(N) C(A, N) \tag{7}$$

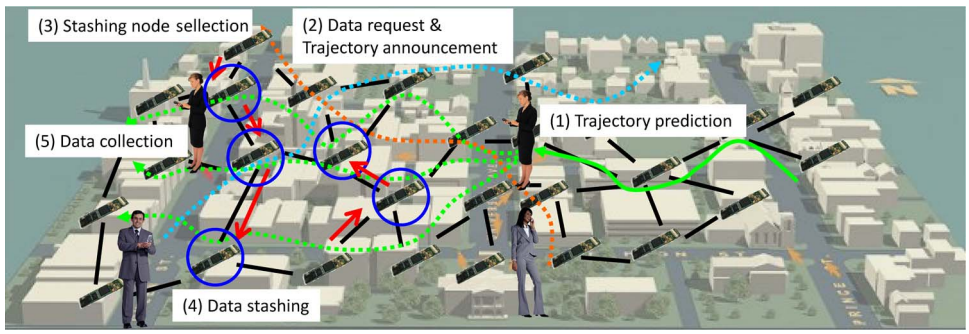


Fig. 6. Overall procedure of our proposed routing protocol of trajectory prediction (by mobile sink), stashing node selection (by data source sensor), and routing.

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

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 $T_i = B_1^i, \ldots, B_{n_i}^i$, we can write this condition as a single linear constraint per trajectory T_i

$$\sum_{0<j\leq n_i} I(B_j^i) \geq 1. \tag{8}$$

Using these definitions, our problem is to find a set R that minimizes (7) subject to the constraints (8). 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.

C. Optimization to Improve Data Latency

We present an extended optimization problem of selecting stashing nodes, considering how long it will take for a mobile node to pick up the data at the stashing nodes. A certain class of applications may require time-sensitive packet delivery to mobile users. To take into account the factor of how far the selected stashing nodes are located currently from mobile nodes in the predicted trajectory, we apply the *regularization* technique [8] by a factor of α. We aim to minimize the weighted sum of the objective functions: 1) the total routing cost from data sources to the selected stashing nodes and 2) the average trajectory distance from mobile nodes to the selected stashing nodes.

We set up a weighted version of an integer program as follows:

$$f' = \sum_N I(N) \cdot \left[C(A,N) + \alpha \frac{\sum_{i=1}^{V} \text{dist}(M_i, N)}{V} \right] \tag{9}$$

where V is the number of mobile nodes, and $\text{dist}(M_i, N)$ is the sequence distance between the mobile node M_i and the stashing node N in the predicted trajectory of M_i.

We find a set of stashing nodes that minimizes the weighted sum of the objective functions (9), while satisfying the same constraints of guaranteeing data delivery on each predicted trajectory, as in (8).

V. Evaluation

We conduct experiments with real-world wireless traces to validate our trajectory clustering algorithm (see Section V-A). We evaluate our data delivery scheme in a real-world test bed (see Section V-B) and a larger network simulation (see Section V-C) by comparing our technique against direct routing that immediately delivers data directly to mobile sinks in terms of routing efficiency and robustness.

We evaluate routing in terms of routing cost, energy saving, 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*) based on [47] where each static sensor node delivers its data to the currently connected static 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)*). We note that all three protocols that we evaluate use the same underlying point-to-point routing protocols, whereas more advanced routing protocols (such as in [12], [13], [40], and [44]) can be integrated with our data stashing algorithm; the evaluation of data stashing benefits for different static protocols is outside the scope of this paper. 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, e.g., because the mobile sink announces them.

When we evaluate the routing cost, we count how many packets were used to deliver data from sensor nodes to destination nodes after the sensors learn the identity of the correct relay or possible relay candidates. In the evaluation of test-bed experiments (see Fig. 9) and larger network simulations (see Figs. 11 and 12), we demonstrate that, even without considering the control cost, our *Stash* scheme requires far fewer data packets than the *Direct* scheme. We also compare energy efficiency of *Direct* and *Stash* schemes. It has been observed that energy consumption for one-off computation tasks in sensor networks is typically dominated by the energy consumed for

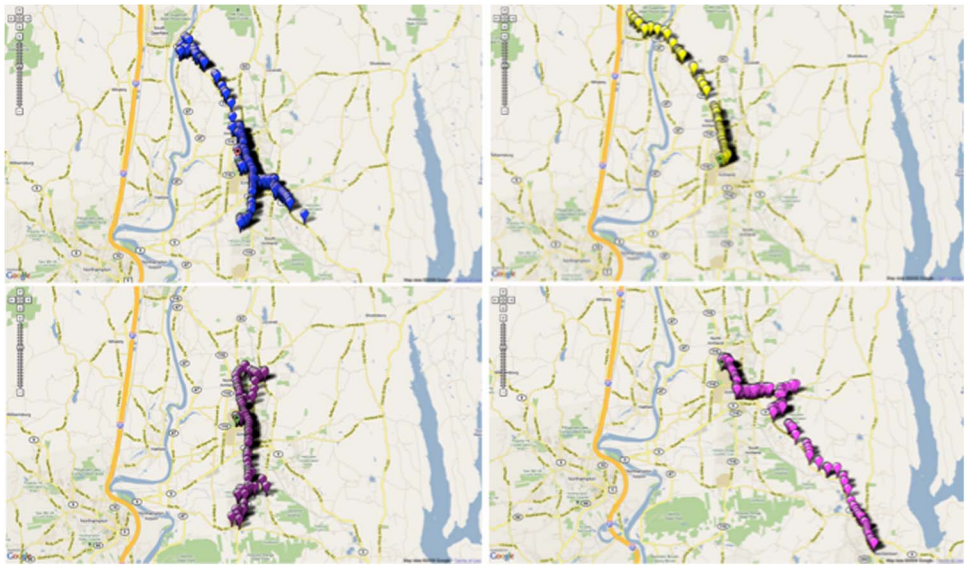


Fig. 7. Typical trajectories of moving buses in UMass from the DieselNet data set. When a bus is associated with a nearby AP, the AP is shown with a marker.

radio transmissions [31], [51]. We thus focus on evaluating energy efficiency in terms of routing cost in our evaluation.

In 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 associated node to the mobile node at any point in time. In reality, even if the stashing node is never selected as the associated 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. We show that benefits of our technique are better load balancing and more even utilization of network resources, such as energy.

Regarding the load balance metric, we measure the number of packets sent by each node and show it in a potential plot and in a cumulative distribution function.

A. Clustering and Trajectory Prediction

First, we validate our probabilistic trajectory model used for prediction using real-world mobility data traces from UMass DieselNet [6] (shown in Fig. 7). The traces consist of time series of wireless AP IDs that wireless cards installed in buses connect to. There are 34 buses, 4198 APs, and 789 bus trips in the data set, covering an area in and around the UMass campus. We evaluate how reliably the selected stashing nodes can connect to mobile sinks. Note that we did not use any bus identification information but used only wireless association list of each bus trip as the input of our trajectory model.

We tested the hierarchical clustering algorithm described in Section III-A. The algorithm ended up with clustering the set of 789 bus trips into 23 clusters. Although we have no ground truth to compare these clusters against, we visually evaluated the clusters and found them to be of good quality.

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 network 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 earlier, 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. As Fig. 8 shows, if we base our predictions on a very short historical trajectory, the prediction quality suffers (*underfitting* problem). For accurate prediction, we need longer series of past movement patterns that will allow us to identify future paths. On the other hand, if the historical trajectory is too long, this constrains the possible future user locations to a small set as few historical trajectories in the training set fit the current data. The error of prediction then increases as a result of *overfitting*. The results show that our prediction method performs well in real-world scenarios for L between 10 and 30.

B. Small-Scale Real-World Network Experiment

We evaluate our algorithms in a real test bed deployed in Clark Center at Stanford University, as in Fig. 9. We used TinyOS 2.1 [1] and configured 41 stationary TelosB sensor nodes [49] to form a sensor network in an 65×100 m^2 area. We set transmission power to 0 dBm that resulted in the radio range of approximately 20–30 m. We asked users to carry a TelosB node in the network along ten different moving paths, as shown Fig. 9. The users moved at a speed of approximately 1.4 m/s, and the mobile node exchanged packets with stationary

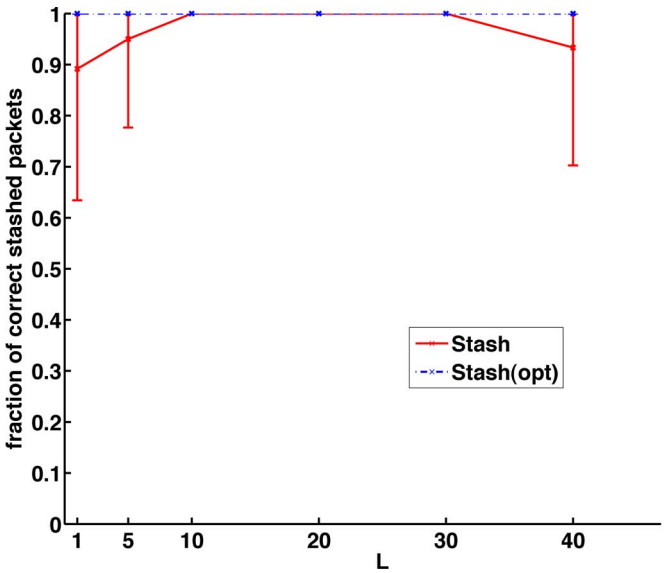


Fig. 8. Fraction of packets stashed on nodes that are actually visited by the mobile node, depending on number of nodes L used for prediction in the DieselNet dataset.

Fig. 9. Forty one stationary sensors (marked with red circle) distributed over 65×100 m^2 and moving paths of mobile sinks in the Clark building at Stanford University. Ten different moving paths, including the opposite direction, are explored while a mobile sink carries a sensor device and communicates with the networks.

nodes at a rate of 1 Hz. The node that replies back to the mobile node with the highest signal strength is considered the association node at every beacon time. In these experiments, all of sensor nodes send data to mobile sinks. It should be noted that each unique moving path is highly overlapped with others in part, and the resulting association nodes are dynamically varying, even with the exact same moving path due to the real wireless vagaries. Any mobile node identification or trajectory information other than associated node IDs is not used. Hence, this experiment setup makes the future trajectory prediction neither obvious nor trivial. For evaluation results, the number of nodes for prediction $L = 10$ is used.

We examine how the number of mobile sinks affects the performance of these algorithms in terms of routing cost, packet delivery reliability, and storage overhead. As the number of mobile sinks increases, routing cost of the *Direct* scheme is proportional to the number of the sinks, whereas stashing algorithms are affected much less because they exploit overlaps in the different trajectories [see Fig. 10(a)]. In terms of packet

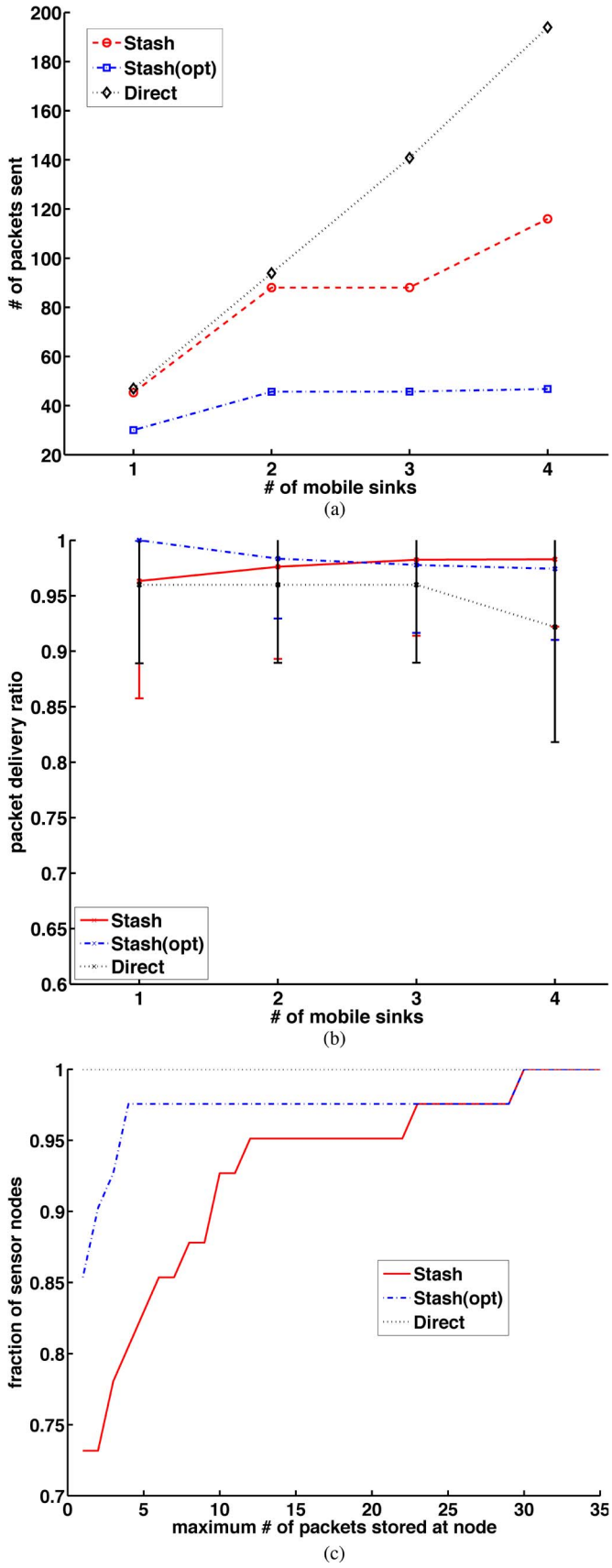


Fig. 10. Routing cost, delivery reliability, and storage cost, depending on the number of mobile sinks in Clark test bed. (a) Routing cost. (b) Reliability. Mean and error bars of standard deviation are shown. (c) Fraction of nodes storing less than a certain number of packets.

delivery, our stashing algorithm achieves high reliability above 95%, whereas the *Direct* scheme suffers due to independent packet delivery directly to each mobile sink while traveling over a larger number of hops [see Fig. 10(b)]. Regarding storage overhead, our *Stash* scheme requires only 10% of sensor nodes to store ten or more packets [see Fig. 10(c)].

C. Large-Scale Network Simulation Experiment

We also test the algorithms in a larger simulated network of downtown San Francisco. The network consists of 716 sensor nodes in an 830×790 m^2 area (see Fig. 11). We generated 20 different trajectories, a subset of which we show in Fig. 12. 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 decibels. These parameters have been derived from measurements in urban environments [20]. We model interference effects using the closest-fit pattern matching model [30] in TinyOS 2.1 [1] with `meyer-light` noise traces.

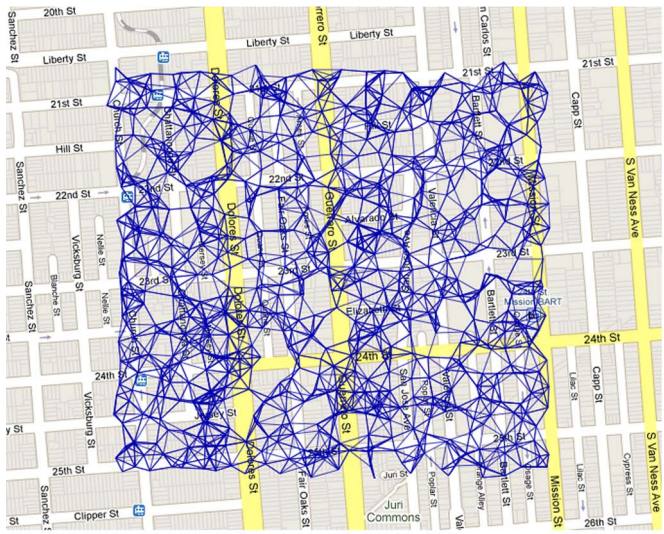
We implemented our routing algorithm in the TinyOS TOSSIM simulator [34] using idealized static shortest-path routing. In our scenario, it is often the case that we route several packets along similar paths. We use multicast to reduce redundant packet transmissions. We ran all of the experiments ten times and draw mean values with standard deviation error bars whenever applicable.

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: the *Direct* scheme requires continuous flooding to announce each mobile sink's current relays, whereas in the *Stash* scheme, the mobile sinks need to announce the anticipated trajectory node IDs only once (unless the network needs restashing for difficult prediction scenarios). In our 716-node topology, it took 682 packet transmissions to disseminate one packet from a mobile sink to the entire network using the Drip dissemination algorithm in TinyOS 2.x. In our simulation setting, the *Direct* method requires one position update every 2 s 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 transmissions 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 s of operation and continuously increases at 341 packet transmissions per second, whereas the overhead for *Stash* remains constant.

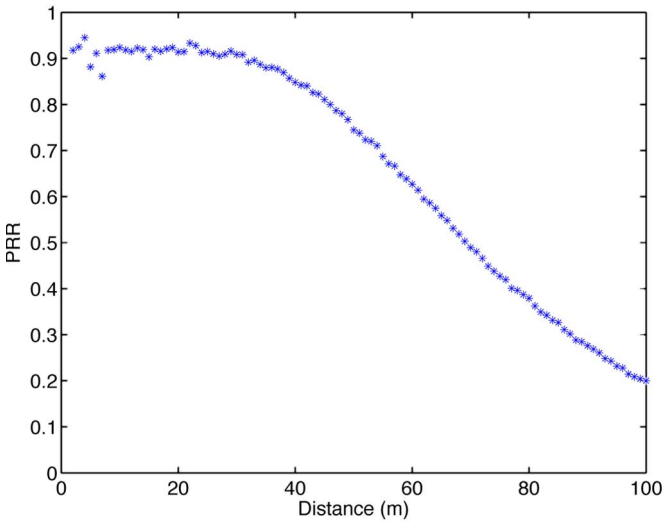
Note that the protocols use global knowledge of the network and deliver data to mobile sinks along shortest routes. A specialized protocol like S4 [40] might be a better choice for the dynamic routing environment in sensor networks. To understand the implications of using a scalable routing protocol 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.

1) Network Performance: We evaluated our network optimization scheme against the direct point-to-point and perfect stashing algorithms using the simulated network. In these



(a)



(b)

Fig. 11. Wireless sensor network in downtown San Francisco, CA, for simulation. A total of 716 sensor nodes are distributed over 830×790 m^2. (a) Connectivity graph over 716 sensor nodes where links are shown for packet reception ratio of $> 75\%$. (b) Wireless connectivity characteristic in simulation.

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

Fig. 12. Moving paths of mobile vehicles where each unique moving path is highly overlapped with others in part. Each vehicle moves at a speed of $\mathcal{N}(30, 5^2)$ in kilometers per hour. We generate 20 different moving paths including the opposite direction as well. All of the 20 vehicles are moving over the networks while communicating with sensor nodes, as in Fig. 11(a).

experiments, all 716 sensor nodes are transmitting data to 1–20 mobile sinks. Given the moving paths of mobile vehicles, as shown in Fig. 12, 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. Although 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. 13(a)]. This effectively prevents network congestion. In fact, data stashing requires only 19% of packets to deliver the same data, compared with direct routing, achieving an energy saving of over 80%. 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. 13(c)]. The *Stash* routing algorithm uses up to 30 retransmissions similar to the state-of-the-art collection tree protocol [19]. 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, This means that the combination of probabilistic prediction and data stashing performs well even under a degree of uncertainty (or prediction error). The *Direct* scheme, on the other hand, requires a large amount of packet transmissions and suffers from poor packet delivery performance. This demonstrates that the *Stash* algorithm can improve routing performance through predictive data dissemination even with a limited knowledge of the future user location.

We also evaluate how the length of predicted trajectories affects performance. If the trajectory prediction is very uncertain and is 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. 14. 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, while significantly increasing energy saving in routing. Note that our optimization scheme can only counterbalance the effects of imperfect trajectory prediction if it is given enough choice. In our experiments, the breakeven 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 increases, each sensor node receives a larger number of anticipated trajectory nodes from mobile sinks and needs to solve a more complex LP. In addition, a larger W means a longer term prediction given the same information about trajectory. In practice, particularly in large networks, where we would expect very long trajectories, one would set a limit of $W \approx 100$.

We investigate the impact of prediction performance with data stashing on packet delivery reliability. The prediction algorithm uses the first L nodes of the sink trajectory to predict the rest. Fig. 15 shows the performance of our prediction algorithm with 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.

We evaluate the timing of packet delivery of each scheme to emphasize why the *Direct* scheme inherently lacks data timeliness. In our simulation setting, the stationary sensor network lead to an average communication hop count of 10, spanning from 1 to 26 hops [the distribution of the number of hops is shown in Fig. 16(a)]. For a mobile sink speed of 30 km/h, the average transition time of mobile nodes is 2.2 s [see the distribution of transition time in Fig. 16(b)]. Fig. 16(c) shows that the *Direct* scheme actually needs much longer than that to send a packet to the association node of the mobile node. This means that when the packet arrives at the destination relay node, the mobile node would likely be out of range already.

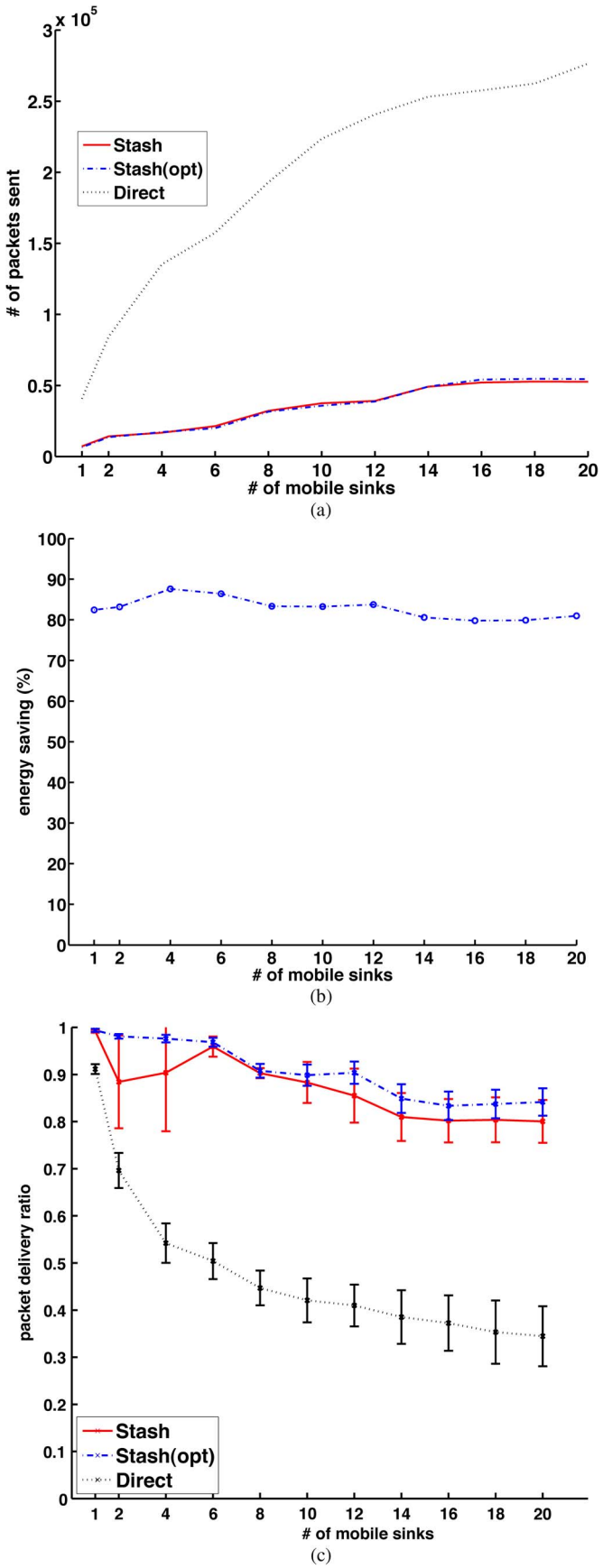


Fig. 13. Routing cost, energy saving, and delivery reliability depending on the number of mobile sinks. (a) Routing cost. (b) Energy saving. (c) Reliability. Mean and error bars of standard deviation are shown.

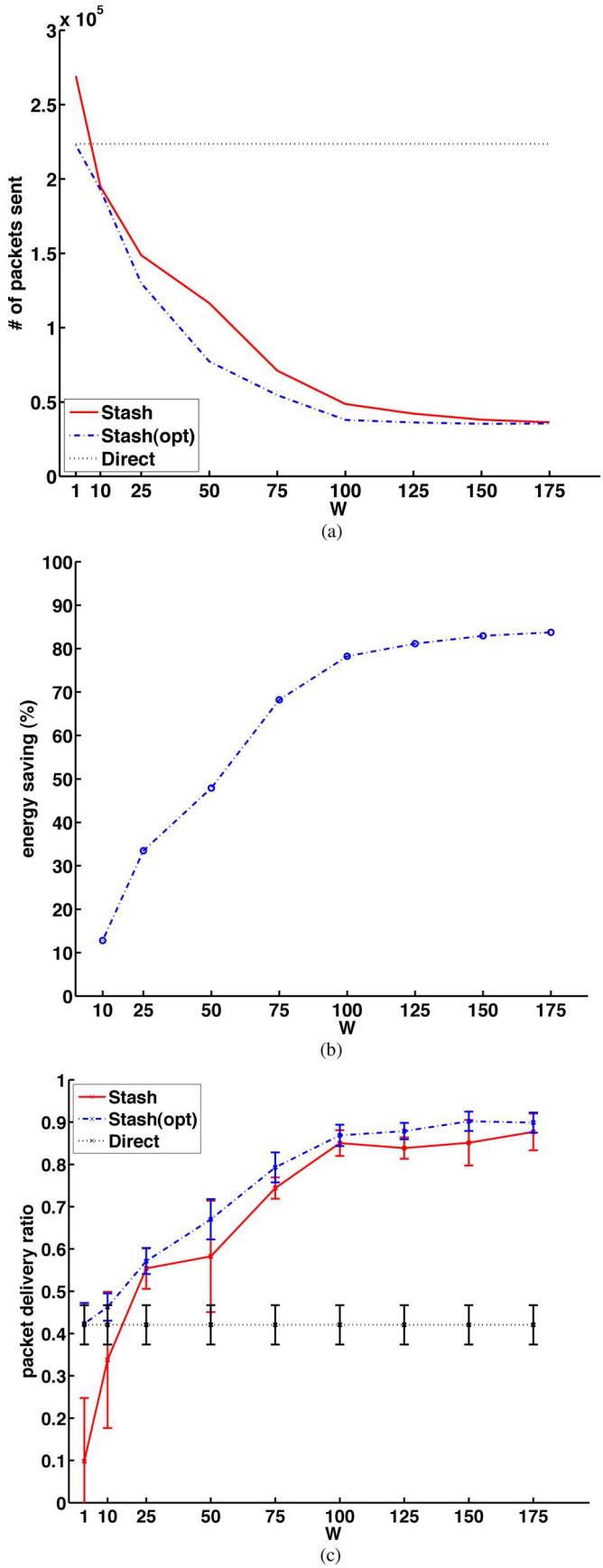


Fig. 14. Routing cost and delivery reliability depending on the number of predicted trajectory nodes W for ten mobile sinks. (a) Routing cost. (b) Energy saving. (c) Packet delivery ratio to mobile sinks, representing the mean value and error bars of standard deviation.

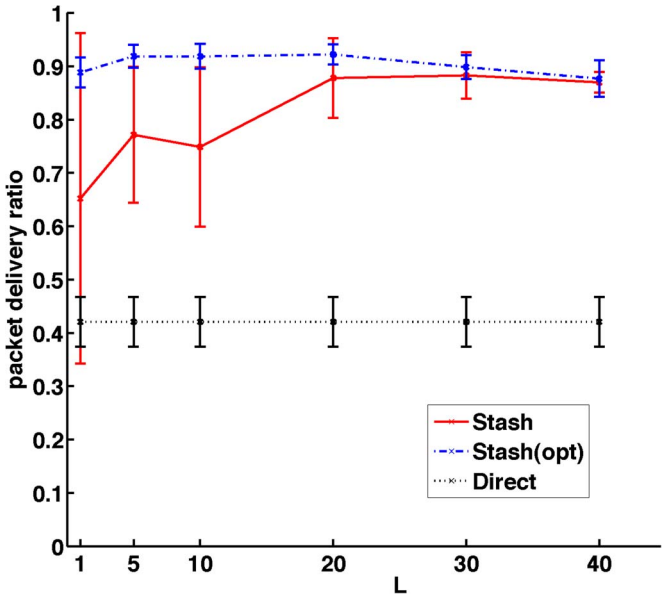


Fig. 15. Packet delivery reliability depending on number of nodes L used for prediction. Data for ten mobile sinks, with mean value and error bars showing standard deviation, is shown.

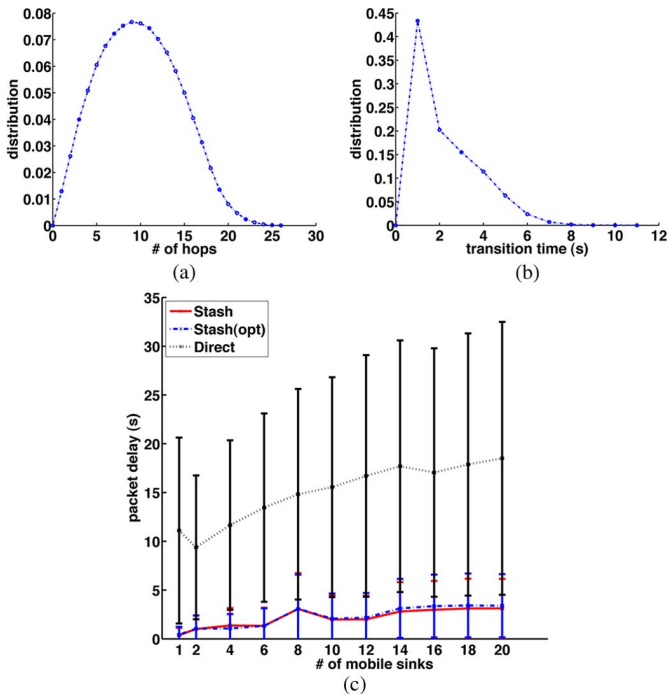


Fig. 16. Distributions of the number of hops and node transition time of mobile sinks in evaluation data and packet delay performance. Large packet delay in the *Direct* scheme would lead to a critical performance degradation in the dynamic transitions of mobile nodes. (a) Distribution of the number of hops throughout the sensor networks. (b) Distribution of sensor node transition time of mobile sinks. (c) Packet delay, representing the mean time and error bars of standard deviation.

In our *Stash* scheme, stashing data at some intermediate storage nodes (somewhere between the data source and the mobile sink) significantly reduces the number of travel hops and, therefore, the packet travel time. Because the intermediate storage node will be visited by the sink in the future, travel time is less of an issue.

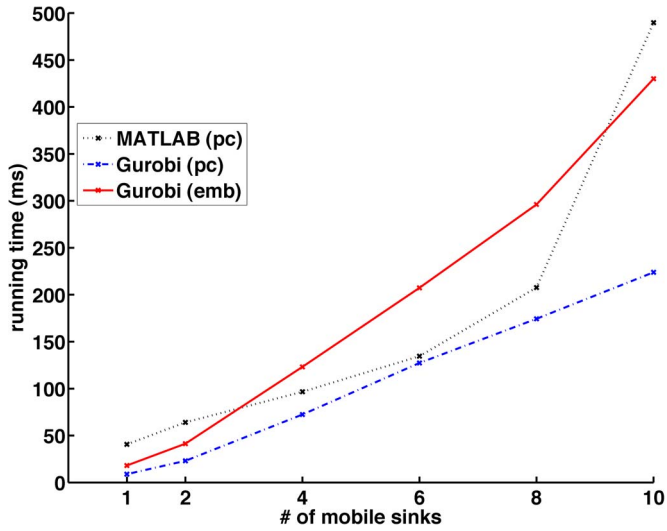


Fig. 17. Running time for a sensor node to solve an optimization problem for stashing in each platform/tool depending on the number of mobile sinks.

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 Section IV-B. The results for different platforms are shown in Fig. 17: We tested the performance on a Dell Precision 390 PC with Ubuntu Linux and a 2.4-GHz Core 2 Duo processor, as well as an embedded platform: a fit-PC2 with Ubuntu Linux and Intel Atom Z530 1.6 GHz. 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.

Another strength of data stashing is implicit load balancing. Fig. 18 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. In the *Direct* scheme, there are many hot regions that transmit a large number of packets [see Fig. 18(a)]; the *Stash* scheme performs much better [see Fig. 18(b)].

We have also tested the robustness of our data stashing scheme against differences in the speed of mobile sinks. Because the trajectory matching algorithm implicitly compensates for speed differences, changes in the speed of mobile sinks do not have a large impact on 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 and 50 km/h and above 70% above for 70 and 90 km/h in Fig. 19.

We explore how the penalty factor α in Section IV-C affects network performance in Fig. 20. As the optimization procedure gives a larger penalty to a set of stashing nodes that are further away from mobile nodes with a larger α, the data pickup time can be improved, as shown in Fig. 20(a). To achieve this benefit, the routing algorithm needs to sacrifice the routing cost as in Fig. 20(b). As the penalty factor increases from 0 to 0.2, the reduced rate of data pickup time is the most drastic, given the similar trend in increasing routing cost.

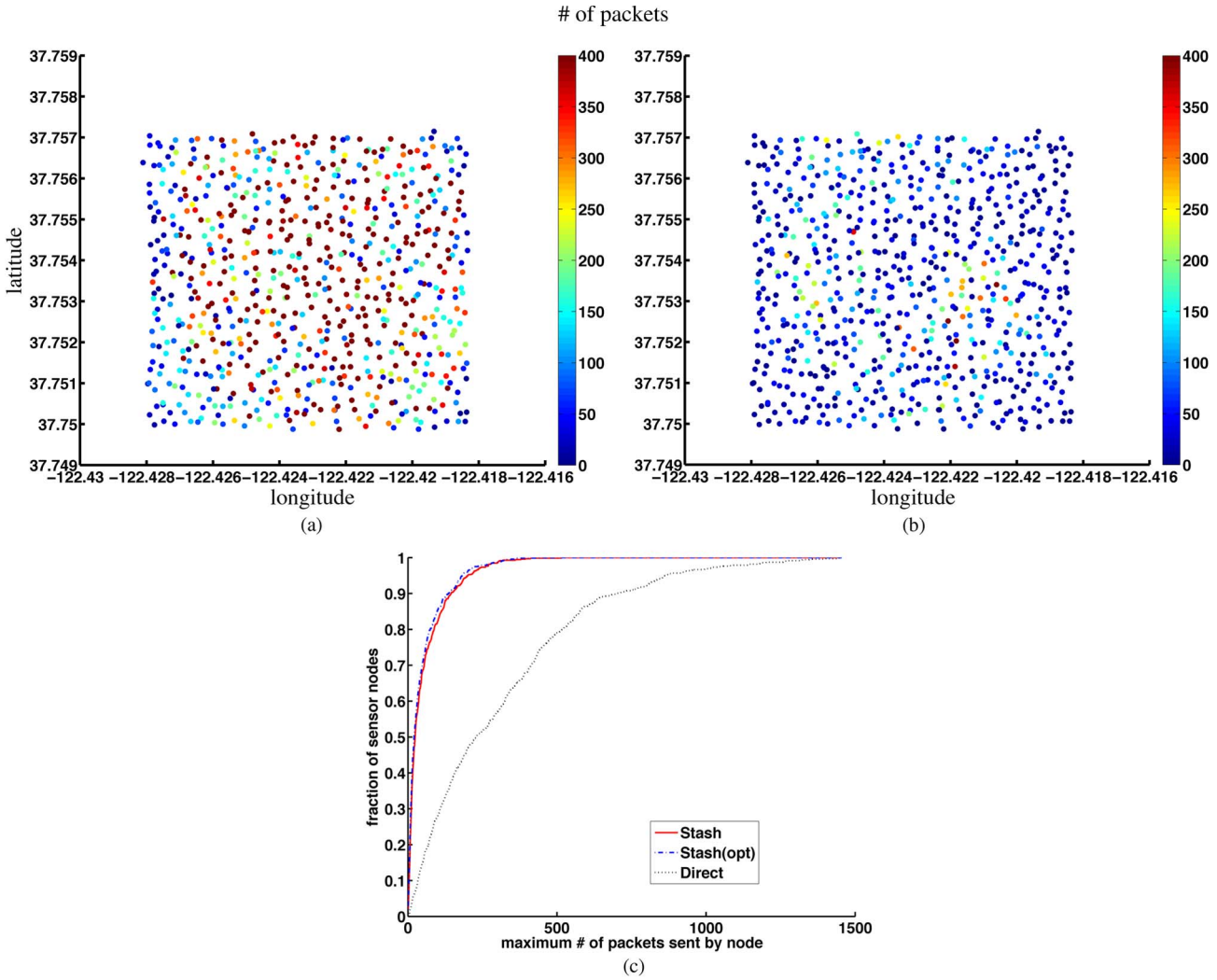


Fig. 18. Load balancing throughout the networks (for the case of ten mobile sinks). (a) Potential plot of the number of packets sent by a node for the *Direct* scheme. (b) Potential plot of the number of packets sent by a node for *Stash* scheme. (c) Fraction of nodes sending less than a certain number of packets.

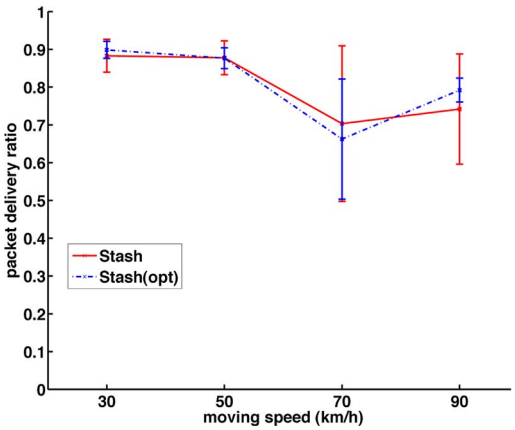


Fig. 19. Packet delivery reliability depending on speed of mobile sinks. Data for ten mobile sinks, mean value, and error bars showing standard deviation are shown.

Finally, we evaluate the storage requirements that data stashing algorithms impose on sensor nodes (see Fig. 21). 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, which turn out to be the intersections of several trajectories, as shown in Fig. 21(b). In our opinion, data storage is generally less problematic than radio transmission in sensor networks, making this a good tradeoff.

VI. Discussions

Here, we try to answer the following questions: 1) How can the data stashing be integrated with duty-cycle MAC protocols; 2) what are differences in empirical results between simulated dataset and real-world data set; and 3) what are the scenarios in which the data stashing scheme may not work well?

A. Integration With Duty Cycle MAC

Although this paper focuses on improving energy efficiency of routing in the network layer, a cross-layer integration with low-duty-cycle MACs will lead to a more significant energy

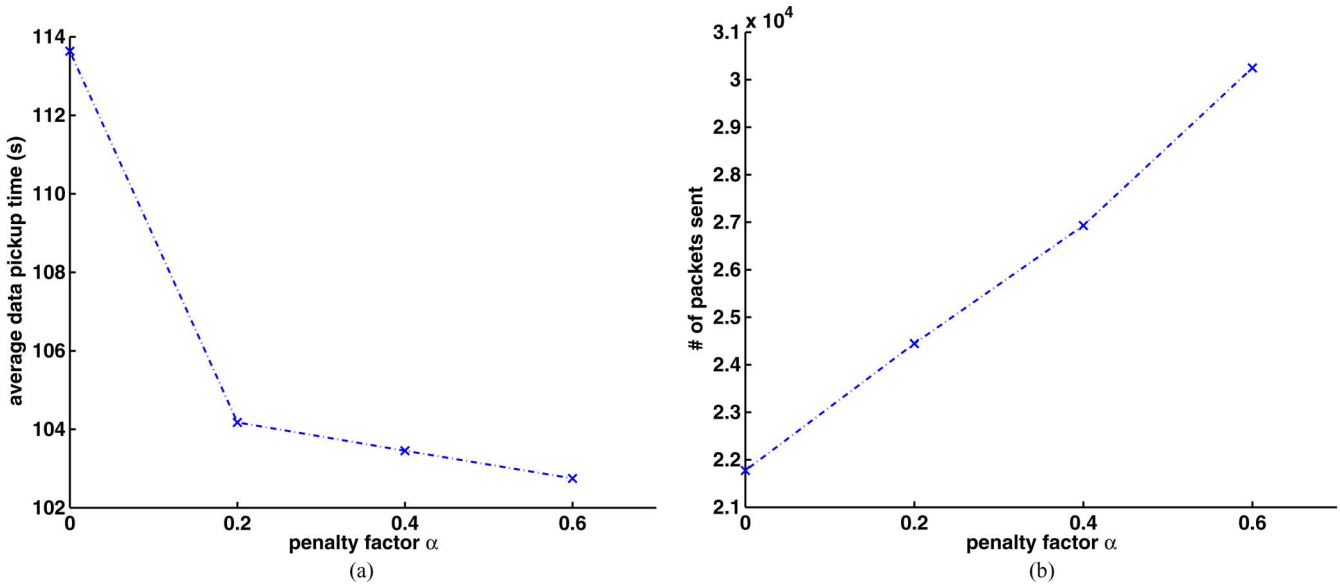


Fig. 20. Impact of the penalty factor α on the data pickup time and the routing cost in the advanced optimization for ten mobile sinks. (a) Average data pickup time by mobile nodes at the stashing nodes with respect to α. (b) Routing cost with respect to α.

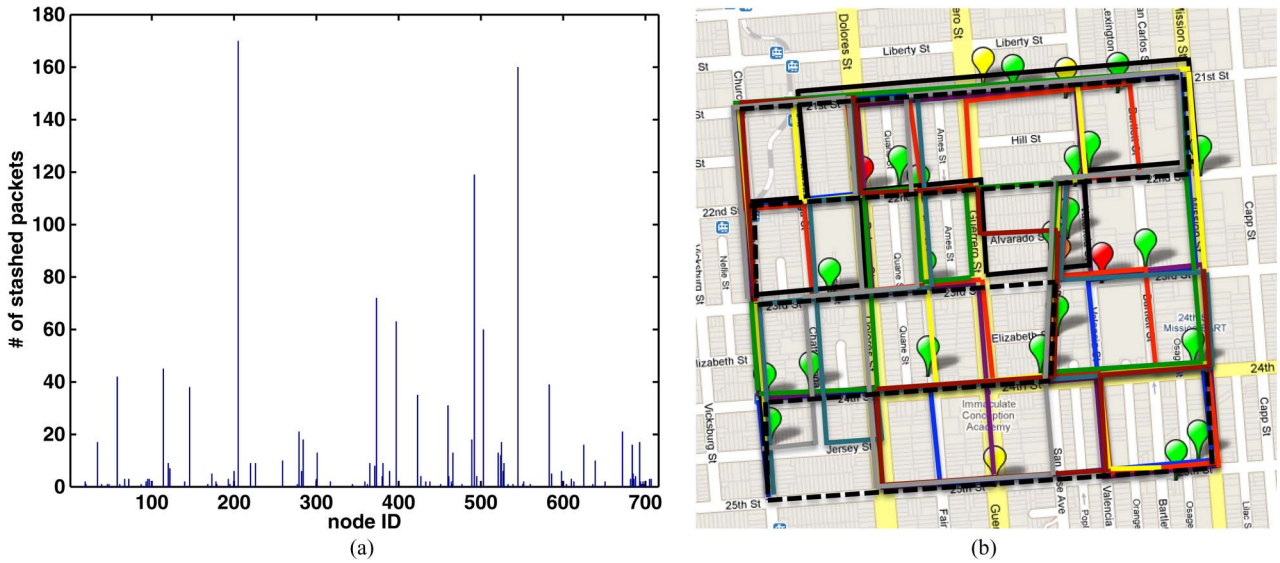


Fig. 21. Storage overhead over the sensor nodes for ten mobile sinks. (a) Storage cost throughout the sensor nodes. (b) Favorite storage node distribution over the networks where ● : $\geq$ 150 packets, ● : $\geq$ 100 packets, ● : $\geq$ 50 packets, and ● : $\geq$ 10 packets, presented with mobile sinks' moving paths.

saving effect. There are two main types of duty cycling MACs in sensor networks: synchronous duty cycle MACs such as S-MAC [66], DMAC [37], Z-MAC [50], and asynchronous duty-cycle MACs such as B-MAC [48] and X-MAC [9]. Our data stashing scheme allows sensor nodes to receive trajectory announcement from a mobile sink at the same time and perform data stashing simultaneously to multiple stashing nodes. For this reason, synchronous MAC protocols are more suitable to data stashing than asynchronous MACs with respect to control overhead for managing all the wake-up schedules.

To integrate the data stashing scheme with synchronous MACs, the optimization problem of stashing node selection needs to be reformulated considering wake-up schedules of sensor nodes. Data source nodes should choose a set of nodes that are only *awake* at the same time for stashing, while keeping a low routing cost. By sorting the nodes awake at a specific time among possible trajectory nodes and simply using them as input in the network optimization (see Section IV-B), it can make our data stashing scheme easily compatible with synchronous duty-cycle MAC protocols.

B. Comparison of Empirical Results Between Simulation and Real-World Evaluation

Both empirical results with simulated and real-world data sets (see Figs. 10 and 13) show similar trends in routing cost and packet delivery reliability with respect to the number of mobile sinks in the network. When we take a closer look at

routing cost results in Figs. 10(a) and 13(a), the reduction of routing cost (between *Direct* and *Stash*) is more substantial in the simulated data set compared with real-world data set as the number of mobile sinks increases. This is related to the variety of stashing node choices from predicted trajectories. Since the Clark test bed forms a relatively small network, the length of wireless traces along the moving paths is smaller compared with wireless traces in the simulation data set. Consequently, finding stashing nodes closer to data sources that are also overlapped with other mobile sinks' future trajectories would be more likely in longer wireless traces. Therefore, our data stashing scheme seems to be more suitable to large-scale networks or densely deployed networks in terms of energy efficiency.

If we compare the routing cost between *Stash* and *Stash(opt)* from simulation and real-world evaluation, the gap between *Stash* and *Stash(opt)* is relatively larger in real-world evaluation compared with simulation. This result is related to prediction quality. Because real-world traces embed a higher variability in association sequences due to more dynamics in wireless vagaries, and walking habit and moving speed variations of humans, the predicted trajectory nodes would be represented with more dynamic probabilities. By selecting a larger number of stashing nodes to balance the guaranteed packet delivery, the real-world evaluation causes a higher routing cost.

C. Limitation in Applicability

One of the most important components in our data stashing scheme is the long-term mobility prediction algorithm that is enabled from intensive pattern learning along moving paths with a certain degree of regularity. If we are given test sequences obtained from new moving paths that do not share most of path segments with paths in the training set, the performance of data stashing would degrade. As an extreme example, our long-term mobility prediction would fail to provide meaningful predictions for test sequences from random motions.

To construct a characteristic database of mobile trajectory clusters, it is important to learn the mobility model from many complete long trips that can be differentiated from one another. If training sequences from only short trips are given in the learning phase, it would harm the quality of our clustering algorithm, making data stashing inefficient.

VII. Conclusion

We have presented algorithms for extracting mobility patterns using association updates over stationary sensor networks, and predicting long-term trajectories of mobile sinks. We focused on the common case that the data are delay tolerant. We have designed a routing scheme that routes data not to the mobile sink directly but, instead, to relay nodes along a predicted path of the mobile sink that are also close to the data source in terms of communication hops. These techniques significantly reduce radio energy consumption for packet routing while ensuring high-packet delivery ratios.

Our experiments indicate that our scheme provides 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. The proposed method provides not only a mobile routing protocol but a way to improve any existing protocol as well 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 restashing at newly predicted nodes could significantly increase robustness.

The trajectory clustering algorithm is currently executed in an offline learning phase. However, our proposed scheme does not necessarily require a separate offline 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 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 multitier or hierarchical approach to deal with extremely large networks is another avenue for future work.

References

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

[2] H. Abou-Zeid and H. S. Hassanein, "Predictive green wireless access: Exploiting mobility and application information," *IEEE Wireless Commun.*, vol. 20, no. 5, pp. 92–99, Oct. 2013.

[3] J. Akbari Torkestani, "Mobility prediction in mobile wireless networks," *J. Netw. Comput. Appl.*, vol. 35, no. 5, pp. 1633–1645, Sep. 2012.

[4] K. Almi'ani, A. Viglas, and L. Libman, "Mobile element path planning for time-constrained data gathering in wireless sensor networks," in *Proc. 24th IEEE Int. Conf. AINA*, 2010, pp. 843–850.

[5] J. A. Alvarez-Garcia, J. A. Ortega, L. Gonzalez-Abril, and F. Velasco, "Trip destination prediction based on past gps log using a hidden markov model," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8166–8171, Dec. 2010.

[6] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine, "Relays, base stations, and meshes: Enhancing mobile networks with infrastructure," in *Proc. ACM MobiCom*, 2008, pp. 81–91.

[7] Z. Becvar, P. Mach, and B. Simak, "Improvement of handover prediction in mobile wimax by using two thresholds," *Comput. Netw.*, vol. 55, no. 16, pp. 3759–3773, Nov. 2011.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sens. Syst.*, 2006, pp. 307–320.

[10] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using predictable observer mobility for power efficient design of sensor networks," in *Proc. IPSN*, 2003, pp. 129–145.

[11] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, "Sink mobility protocols for data collection in wireless sensor networks," in *Proc. 4th ACM Int. Workshop Mobility Manag. Wireless Access*, 2006, pp. 52–59.

[12] J. Chroboczek, "The babel routing protocol," IETF, Fremont, CA, USA, RFC 6126, Apr. 2011.

[13] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF, Fremont, CA, USA, 2003.

[14] R. Coltun, D. Ferguson, and A. L. J. Moy, "OSPF for IPv6," IETF, Fremont, CA, USA, RFC 5340, 2008.

[15] G. E. Crooks, G. Hon, J.-M. Chandonia, and S. E. Brenner, "WebLogo: A sequence logo generator," *Genome Res.*, vol. 14, no. 6, pp. 1188–1190, Jun. 2004.

[16] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM TOSN*, vol. 8, no. 1, pp. 1–34, Aug. 2011.

[17] D. Feng *et al.*, "A survey of energy-efficient wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 167–178, First Quarter 2013.

[18] J. Ghosh, M. Beal, H. Ngo, and C. Qiao, "On profiling mobility and predicting locations of campus-wide wireless network users," Tech. Rep., State Univ. New York, Buffalo, NY, USA, Jan. 2005.

[19] O. Gnawali *et al.*, "Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM TOSN*, vol. 10, no. 16, pp. 1–16, Nov. 2013.

[20] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge Univ. Press, 2005.

[21] Green ICN. [Online]. Available: http://www.greenicn.org

[22] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw.*, 2000, pp. 56–67.

[23] U. Javed *et al.*, "Predicting handoffs in 3G networks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 45, no. 3, pp. 65–70, Dec. 2011.

[24] D. Johnson, D. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Proc. Ad Hoc Netw.*, 2001, pp. 139–175.

[25] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *Proc. ACM SenSys*, 2003, pp. 193–204.

[26] D. Kotz, T. Henderson, and I. Abyzov, CRAWDAD Data Set Dartmouth/Campus (v. 2004-12-18), Dec. 2004. [Online]. Available: http://www.crawdad.org/dartmouth/campus

[27] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *J. Location Based Serv.*, vol. 7, no. 2, pp. 98–120, Jun. 2013.

[28] B. Kusy, H. Lee, M. Wicke, N. Milosavljevi'c, and L. Guibas, "Predictive QoS routing to mobile sinks in wireless sensor networks," in *Proc. IPSN*, 2009, pp. 109–120.

[29] N. D. Lane *et al.*, "Enabling large-scale human activity inference on smartphones using Community Similarity Networks (CSN)," in *Proc. 13th Int. Conf. Ubiquitous Comput.*, 2011, pp. 355–364.

[30] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Proc. 6th Int. Symp. IPSN*, pp. 21–30, 2007.

[31] H. Lee, H. Kim, and I. J. Chang, "CPAC: Energy-efficient data collection through adaptive selection of compression algorithms for sensor networks," *Sensors*, vol. 14, no. 4, pp. 6419–6442, Apr. 2014.

[32] H. Lee, M. Wicke, B. Kusy, O. Gnawali, and L. Guibas, "Data stashing: Energy-efficient information delivery to mobile sinks through trajectory prediction," in *Proc. 9th ACM/IEEE IPSN*, 2010, pp. 291–302.

[33] Y.-S. Lee and S.-B. Cho, "Human activity inference using hierarchical bayesian network in mobile contexts," in *Proc. Neural Inf. Process.*, 2011, pp.38–45.

[34] P. Levis, N. Lee, M. Welsh, and D. Culler "TOSSIM: Simulating large wireless sensor networks of TinyOS motes," *Proc. ACM SenSys*, 2003 pp.126–137.

[35] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings Bioinformat.*, vol. 11, no. 5, pp. 473–483, Sep. 2010.

[36] Y. Liu and B. Schmidt, "Multiple protein sequence alignment with msaprobs," in *Proc. Methods Molecular Biol.*, 2014, pp. 211–218.

[37] G. Lu, B. Krishnamachari, and C. S. Raghavendra "An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks," in *Proc. 18th Int. Symp. Parallel Distrib.*, 2004.

[38] M. Ma, Y. Yang, and M. Zhao, "Tour planning for mobile data-gathering mechanisms in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1472–1483, May 2013.

[39] G. Malkin, RIP, ver. 2, 1998.

[40] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. Smith, "S4: Small state and small stretch compact routing protocol for large static wireless networks," *IEEE/ACM TON*, vol. 18, no. 3, pp. 761–774, Jun. 2010.

[41] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdisciplinary Rev., Data Mining Knowl. Discovery*, vol. 2, no. 1, pp. 86–97, Jan./Feb. 2012.

[42] A. Nadembega, A. Hafid, and T. Taleb, "A path prediction model to support mobile multimedia streaming," in *Proc. IEEE ICC*, 2012, pp. 2001–2005.

[43] A. Nadembega, T. Taleb, and A. Hafid, "A destination prediction model based on historical data, contextual knowledge and spatial conceptual maps," in *Proc. IEEE ICC*, 2012, pp. 1416–1420.

[44] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to ad-hoc networking (batman) draftwunderlich-open-mesh-manet-routing-00," Network Working Group, 2011.

[45] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 551–591, Second Quarter 2013.

[46] 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, Nov. 2001.

[47] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM SIGCOMM*, vol. 24, no. 4, pp. 234–244, Oct. 1994.

[48] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sens. Syst.*, 2004, pp. 95–107.

[49] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. 4th Int. Symp. IPSN*, 2005, pp. 364–369.

[50] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-mac: A hybrid mac for wireless sensor networks," *IEEE/ACM TON*, vol. 16, no. 3, pp. 511–524, Jun. 2008.

[51] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proc. 4th Int. Conf. Embedded Netw. Sens. Syst.*, 2006, pp. 265–278.

[52] K. Samdanis and M. Schoeller, "Exploiting user context information for energy management in enterprise femtocell networks," in *Proc. IEEE/IFIP Int. Symp. Integrated Netw. Manag.*, May 2013, pp. 361–368.

[53] C. A. Santiváñez, R. Ramanathan, and I. Stavrakakis, "Making link-state routing scale for ad hoc networks," in *Proc. MobiHoc*, 2001, pp. 22–32.

[54] I. F. Senturk and K. Akkaya, "Mobile data collector assignment and scheduling for minimizing data delay in partitioned wireless sensor networks," in *Proc. Ad Hoc Netw.*, 2014, pp. 15–31.

[55] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Molecular Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.

[56] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.

[57] L. Song, U. Deshpande, U. Kozat, D. Kotz, and R. Jain, "Predictability of WLAN mobility and its effects on bandwidth provisioning," in *Proc. 25th IEEE INFOCOM*, 2006, pp. 1–13.

[58] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," in *Proc. 23rd Annu. Joint Conf. IEEE INFOCOM*, 2004, pp. 1414–1424.

[59] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal W," *Nucl. Acids Res.*, vol. 22, no. 22, pp. 4673–4680, Nov. 1994.

[60] W. Wanalertlak *et al.*, "Behavior-based mobility prediction for seamless handoffs in mobile wireless networks," *Wireless Netw.*, vol. 17, no. 3, pp. 645–658, Apr. 2011.

[61] R. Wohlers, N. Trigoni, R. Zhang, and S. Ellwood, "Twinroute: Energy-efficient data collection in fixed sensor networks with mobile sinks," in *Proc. 10th MDM*, 2009, pp. 192–201.

[62] L. Xiang, J. Luo, and C. Rosenberg, "Compressed data aggregation: Energy-efficient and high-fidelity data collection," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1722–1735, Dec. 2013.

[63] L. Xiang, J. Luo, and A. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," in *Proc. 8th Annu. IEEE Commun. SECON*, 2011, pp. 46–54.

[64] R. V. Yampolskiy and A. El-Barkouky, "Wisdom of artificial crowds algorithm for solving np-hard problems," *Int. J. Bio-Inspired Comput.*, vol. 3, no. 6, pp. 358–369, Nov. 2011.

[65] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proc. ACM MobiCom*, 2002, pp. 148–159.

[66] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[67] Q. Yuan, I. Cardei, and J. Wu, "An efficient prediction-based routing in disruption-tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 19–31, Jan. 2012.

[68] L. Zhao and A. Y. Al-Dubai,"Routing metrics for wireless mesh networks: A survey," *Recent Adv. Comput. Sci. Inf. Eng.*, pp. 311–316, 2012.

[69] M. Zhao, M. Ma, and Y. Yang "Efficient data gathering with mobile collectors and space-division multiple access technique in wireless sensor networks," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 400–417, Mar. 2011.

HyungJune Lee (M'13) received the B.S. degree from Seoul National University, Seoul, Korea, in 2001 and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, USA, in 2006 and 2010, respectively, all in electrical engineering.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Ewha Womans University, Seoul. He joined Broadcom as a Senior Staff Scientist, working on research and development of 60-GHz 802.11ad system-on-a-chip medium-access control. He worked for AT&T Labs as a Principal Member of Technical Staff with the involvement of Long-Term Evolution (LTE) overload estimation, LTE Wi-Fi interworking, and heterogeneous networks. His current research interests include future wireless networks over Internet of Things, 60-GHz, fifth-generation cellular, and heterogeneous networks.

Martin Wicke received the Ph.D. degree from ETH Zurich, Zurich, Switzerland, working on computer animation and physical simulations.

He worked on sensor networks while at Stanford University, Stanford, CA, USA. He is currently working on artificial intelligence and programming languages at eddy.systems, San Francisco, CA.

Branislav Kusy (M'11) received the Ph.D. degree from Vanderbilt University, Nashville, TN, USA.

He spent two years at Stanford University, Stanford, CA, USA, as a Postdoctoral Fellow. He is currently a Principal Research Scientist with Autonomous Systems, CSIRO, Pullenvale, QLD, Australia. He has more than ten years of experience with developing system services for networked embedded systems, algorithms for spatiotemporal coordination of nodes, and real-world deployments of sensor network technology. He is currently focusing on innovative signal processing and adaptive sampling in distributed embedded networks operating under various constraints, such as uncontrolled mobility, limited energy resources, or the lack of physical access to the device. Specifically, he is working with wearable technology for tracking location, activity, and environmental factors for various applications, including disease spread in animal/human populations, biosecurity, and future manufacturing.

Omprakash Gnawali (M'09) received the Master's and Bachelor's degrees from Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA.

He was a Postdoctoral Scholar with Stanford University, Stanford, CA, USA. He is currently an Assistant Professor with the University of Houston, Houston, TX, USA. His contribution to wireless sensor network routing, i.e., collection tree protocol, has been widely adopted in research and academia and has improved the IPv6 low-power wireless networking standard. His research interests include the intersection of low-power wireless networks and embedded sensing systems.

Leonidas Guibas (F'12) received the Ph.D. degree from Stanford University, Stanford, CA, USA, under the supervision of D. Knuth.

His main subsequent employers were Xerox PARC; DEC/SRC; the Massachusetts Institute of Technology, Cambridge, MA, USA; and Stanford University, Stanford, CA, USA. He is currently the Paul Pigott Professor of Computer Science (and by courtesy, electrical engineering) with Stanford University. He heads the Geometric Computation group and is part of the Graphics Laboratory, the AI Laboratory, the Bio-X Program, and the Institute for Computational and Mathematical Engineering. His research interests include geometric data analysis, computational geometry, geometric modeling, computer graphics, computer vision, robotics, ad hoc communication and sensor networks, and discrete algorithms. Some of his well-known past accomplishments include the analysis of double hashing, red–black trees, the quad-edge data structure, Voronoi–Delaunay algorithms, the Earth Mover's distance, Kinetic Data Structures, Metropolis light transport, and the heat-kernel signature.

Dr. Guibas is a Fellow of the Association for Computing Machinery (ACM). He received the ACM Allen Newell Award.