This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.


# The Boomerang Protocol: Tying Data to Geographic Locations in Mobile Disconnected Networks

Tingting Sun Bin Zan Yanyong Zhang Marco Gruteser
WINLAB, Rutgers University
Technology Center of New Jersey
671 Route 1 South
North Brunswick, NJ 08902-3390
{sunting,zanb,yyzhang,gruteser}@winlab.rutgers.edu

***Abstract*—We present the *boomerang protocol* to efficiently retain information at a particular geographic location in a sparse network of highly mobile nodes without using infrastructure networks. To retain information around certain physical location, each mobile device passing that location will carry the information for a short while. This approach can become challenging for remote locations around which only few nodes pass by. To address this challenge, the boomerang protocol, similar to delay-tolerant communication, first allows a mobile node to carry packets away from their location of origin and periodically returns them to the anchor location. A unique feature of this protocol is that it records the geographical trajectory while moving away from the origin and exploits the recorded trajectory to optimize the return path.**

**Simulations using automotive traffic traces for a southern New Jersey region show that the boomerang protocol improves packet return rate by 70% compared to a baseline shortest path routing protocol. This performance gain can become even more significant when the road map is less connected. Finally, we look at adaptive protocols that can return information within specified time limits.**

***Index Terms*—Geocache, GPS, Mobile, Location-aware information, Infrastructureless data management**

## I. Introduction

As daily mobile devices such as smart phones, PDAs and digital cameras start to be used as sensing devices [1]–[3], mobile sensing is becoming a social event instead of a high-tech phenomenon. Compared to today's special-purpose sensing applications such as automotive traffic congestion monitoring [4] and pothole detection [5], mobile sensing takes place anytime, anywhere, and will have far more diverse meanings. A direct consequence of this trend is the production of a vast amount of data, in terms of both type and volume. Example data types include pictures, videos, audios, and plain text-based sensor readings. These data can potentially bring great convenience to the society as they can serve as traces of our lives and logs of the physical world.

Fully utilizing these data, however, demands the establishment of channels between data producers and consumers. We have seen several methods that were used to establish such channels in earlier systems. In many web applications, data producers upload their data to servers, and consumers can either directly contact the server or locate the server through a search engine; in many peer-to-peer data sharing applications, directories are used to map data names to their locations. Though these methods have proven success in their intended systems, they are unsuitable for the anytime-anywhere personal sensing. In personal sensing, there is no fixed relationship between data producers and consumers. Data is more likely to be produced unintentionally than purposefully, and the value of the data is discovered post-facto. Consequently, we may end up having much more data than what will be needed later, and uploading these data can place a huge burden on the underlying network. In addition, privacy can be a serious concern in a server-centric solution as well. This relationship, i.e. having many more producers than consumers, is opposite from what we have observed in other systems, and thus calls for a new data sharing architecture.

To address this challenge, we take inspiration from real life solutions. Suppose if we lost/found an item, a common practice is to post a note around the area where it was lost/found, and later we refer back to the same location to check for further updates. Similarly, in the anytime-anywhere mobile sensing era, information is commonly tagged with location, thus encouraging location-based queries. To facilitate such queries, we advocate building "directories" around locations of interest by having nearby mobiles carry the data (or the metadata of these data) generated around these locations. We refer to the directory information as the *Geocache* of the location[1], and the location of interest as *anchor location*. By always having the node close to the anchor location carry the Geocache, we can tie the data around the location where they were generated, thus easily facilitate location-based queries by directing them to the corresponding anchor locations using any of the geo-routing [6] or geocasting [7] [8] techniques. Once the Geocache for an anchor location reaches a certain size, we have the options of compressing the data, or applying the "chaining" technique, which retains only the latest Geocache entries around the anchor location while saving a link to the storage of older entries. Finally, we may also delete outdated or trivial entries.

[1]Inspired by physical Geocaches that store information and items at specific locations. Finding them with GPS receivers has become a popular pastime (http://en.wikipedia.org/wiki/Geocaching).

Digital Object Indentifier 10.1109/TMC.2011.132 

In this paper, we study protocols that retain Geocache around the anchor location through inter-vehicle communication. Specifically, we address two major challenges: (i) returning the Geocache to the anchor location with high probability if the carrier of the Geocache becomes temporarily disconnected; (ii) minimizing the communication overhead for retaining the Geocache near an anchor location.

The boomerang protocol addresses these challenges by using a trajectory-based approach. It increases the successful return probability of the Geocache even in temporary disconnected scenarios. While the boomerang protocol is inspired by delay-tolerant geographic routing, it is unique in recording a node's trajectory as the node is moving away from the anchor location and using this trajectory as a guidance to carry back the Geocache. Further, to reduce communication overhead, instead of each node sending the Geocache over the wireless link as soon as it was received, we have the node keep the Geocache until it drives off the original trajectory. Thus, it exploits an important characteristic of vehicular networks, which is: vehicles move on well-defined and usually bidirectional paths. We will show through analysis and simulations how this characteristic impacts the performance. In connected networks, the increased return probability allows significantly reduced communication overhead by purposefully allowing a node to briefly carry the information away from the anchor location before returning it, instead of constantly keeping the Geocache at the anchor location.

In summary, the salient contributions of our work are:

- Outlining the Geocache concept, which can be used to make sensed data available at anchor locations, to support mobile sensing applications over a distributed network of mobile nodes.
- Designing the boomerang protocol which periodically return the Geocache to its anchor location. The protocol employs two alternative heuristics in selecting Geocache carriers, with the baseline approach based on a node's distance to the anchor location, and the improved approach based on a node's location relative to the Geocache's reverse trajectory.
- Showing through simulations using the traffic trace for a southern New Jersey area, that the use of trajectory in the boomerang protocol increases the Geocache return probability by an average of $70\%$ compared to the baseline shortest distance routing approach.
- Introducing the parameter $\rho$ that defines the connectivity of a road map, and showing through analysis that the trajectory-based boomerang protocol outperforms the baseline distance-based boomerang protocol under realistic $\rho$ values.
- Designing protocols that return the Geocache within a specified time constraint through Q-learning.

The remainder of the paper is organized as following: Section II briefly discusses the platform assumption and system model. Section III describes in detail the boomerang protocol and Section IV discusses its implementation, especially the techniques used for detecting divergence from a recorded trajectory based on GPS traces. Section V analytically evaluates the boomerang protocol's performance on a Manhattan grid. Section VI compares the performance of the two boomerang protocol variants using real world road maps and traffic patterns. Section VII extends the protocol to support applications with return time constraints. Section VIII discusses the related work, and Section IX concludes the paper.

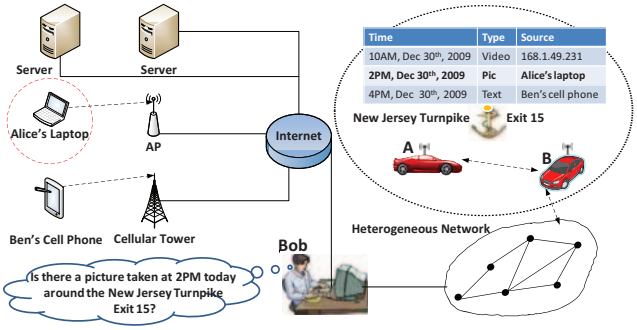



Fig. 1. Motivating scenario of Geocache.

## II. System Model and Applications

We consider a scenario where nodes move along constrained two-way paths. Nodes can communicate intermittently via high-bandwidth short-range radios (e.g., 802.11) with other nearby nodes or through a continuous low-bandwidth wide-area network (e.g., cellular network). We assume that nodes have high storage capacity and are aware of their geographic positions (e.g., via GPS), but the communication and localization systems do not rely on proprietary road maps.

**Mobile Data Management Through Geocache:** As mobile devices start to produce large volumes of data, efficient management of such data can bring great convenience to our daily life. Let us look at the motivating scenario illustrated in Fig. 1.

Alice took a picture of a car accident using her cell phone when she drove by the accident scene. Bob, the victim in the accident, was eagerly seeking such pictures as evidence to support his claims in front of the judge or his insurance company. A traditional solution for exchanging the information would likely involve Alice uploading her picture to a server where Bob can download from (after consulting popular search engines such as Google). However, as mentioned earlier, this solution does not scale well with the data volume we may expect from anytime-anywhere mobile sensing. Instead, we propose a highly distributed approach in which mobile devices keep the data locally, but leave a log around the geographic location where the data was generated. A typical log may include the time and location at which the data was generated, the data type (video, audio, picture, etc), and an (encrypted) ID of the mobile device. In this approach, logs generated around

the same location from different mobile devices will form a location-based log file, which we call *Geocache*, and it will be retained around the anchor location by surrounding mobiles ($A$ and $B$ in Fig. 1). The Geocache serves the same purpose as the bulletin board in our daily lives. Queries such as "Is there a picture taken at 2PM today around the New Jersey Turnpike Exit 15?" will be routed to the corresponding anchor location to look up the Geocache and find out whether related information exists. If matching traces are found, the data requester may directly contact the data producers, and possibly paying a fee for the retrieved data. Based on the same idea, range queries can also be enabled in which queries concerning wider areas will be sent to multiple anchor locations.

A lot of issues need to be carefully evaluated in implementing this architecture. To name just a few, a Geocache may quickly become too large for a popular anchor location, a Geocache may get lost for a remote anchor location that few mobiles access, a Geocache may receive so many requests that it incurs a large delay for an urgent request, etc. In this paper, we set out to attack the most important challenge: *how can a Geocache be retained around its anchor location by passing by mobile nodes in an efficient way?*

## III. Geocache Anchoring Protocols

The goal of the Geocache anchoring protocols is to retain Geocache data around the corresponding anchor location while minimizing communication overhead.

Intuitively, we envision the following anchoring process: the mobile node that currently carries the Geocache (referred to as the *carrier*) is moving away from the anchor location. To avoid taking the Geocache away, it hands off the data to other nodes, preferably those traveling towards the anchor location. After receiving the data, the new carrier node will periodically examine whether another handoff is needed. This process repeats until the data returns to the anchor location, and we call this protocol a *boomerang* protocol because the data eventually returns to its origin like a boomerang.

To motivate how this boomerang approach can reduce communication overhead, let us consider a brief thought experiment. One could retain information at the anchor location by simply handing off the Geocache whenever the anchor location moves out of the radio range. In an idealized model with constant radio range $r$, vehicular velocity $v$ and high vehicle density, retaining the Geocache for a duration $t$ would require $m = \frac{tv}{r}$ handoffs. However, under ideal settings, the boomerang approach can reduce the number of transmissions to $m = 2$ (one to the new carrier heading back and one to the anchor location). Thus, the boomerang approach has the potential to significantly reduce transmission overhead when the Geocache content is only needed periodically. Under more realistic settings, the number of transmissions in the boomerang protocol may be larger because the chosen carrier may diverge at intersections from the original path and not return to the anchor location. The vehicles may also need to send periodic broadcast messages to identify the Geocache for the same anchor location on other nodes and enable operations such as aggregation and update.

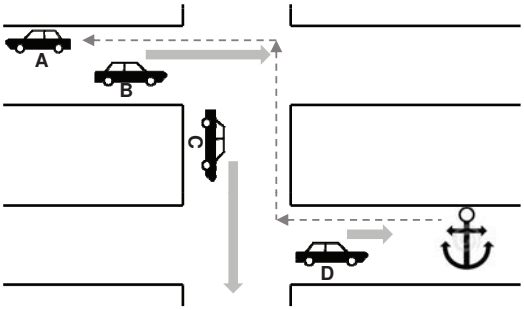



Fig. 2. Sometimes, a single carrier node is insufficient to anchor the data. In this example, after $A$ hands off the data, we need $B$, $C$, $D$ to return the data to its anchor.

Some optimization techniques can be used to further reduce the size of the Geocache. First of all, we can compress Geocache entries. Similar logs from the same mobile can be compressed into one entry. Even logs from multiple mobiles can be compressed for spatial efficiency. Second of all, if the geocache fills up, we can employ a "chaining" technique wherein the geocache only contains the latest log entries (or the most relevant log entries based upon the replacement algorithms). It also contains the ID of the mobile that has the less recent entries, so that when needed later, the mobile can be located via its ID for the geocache page it's carrying.

### A. Protocol Description

The main challenge for implementing the boomerang protocol lies in the choice of a new carrier node at each handoff, especially if the first handoff occurs somewhere far away from the anchor location. The data may have traveled along a rather complicated route before the current carrier looks for a new carrier, as illustrated in Fig. 2. In this case, a single carrier node may not be sufficient to bring back the data; instead, nodes $B$, $C$, and $D$ all needed to be involved in this returning process. Efficiently choosing a set of suitable carriers is thus the key to the success of the boomerang protocol. A set of poorly-selected carriers may incur a long delay in bringing back the data (note that the data may lose its value after a long delay). The task of choosing appropriate carrier nodes is particularly daunting because at each handoff, neither the current carrier nor the nodes within the hand off range have knowledge beyond their current velocity and location, and the traversed trajectory.

Another challenge is the handoff criteria. When to hand off is a tricky issue, especially at the first time. The first handoff can greatly impact the handoff frequency (and thus communication overhead) and return probability of the Geocache. Recognizing the importance of this problem, we dedicated Section VII to study this issue. The rest of the handoffs are easier to decide. In this paper, we propose a trajectory-based carrier selection approach and compare it to a baseline

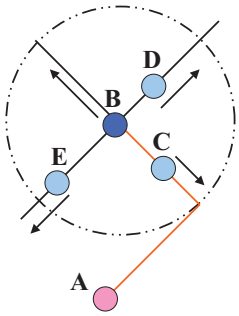



Fig. 3. An example handoff situation. In this case, $B$ is the current carrier. MaxProgress will choose node $E$ as the new carrier (because its distance from A is decreasing and it is currently the closest to A among all the nodes), while RevTraj will choose $C$ as the new carrier.

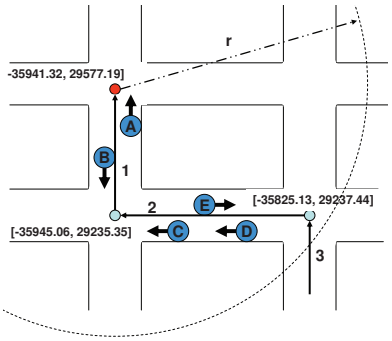



Fig. 4. Illustration of segments and trajectory-based handoff procedure.

shortest-distance-based selection scheme, and we'll discuss the handoff criteria for both schemes in the following section.

**Shortest-Distance-Based Selection:** Heuristics that fall in this category choose the node, among all those within the handoff range, that is closest to and moving towards the anchor location. They share the same rationale as many Geo-routing algorithms [9], and we consider such an algorithm which we refer to as *MaxProgress*. A simple example is given in Fig. 3. $A$ is the anchor location, and $B$ is the carrier node. At handoff, $E$ will be chosen as the next carrier node because its distance from $A$ is the shortest among all other nodes within radio range and still decreasing.

Next, let us look at the detailed handoff procedure for MaxProgress. After traveling away from the anchor location for a certain amount of time, the carrier initiates a handoff by first broadcasting the Geocache along with the anchor location. Every node within the handoff range responds by checking its distance from the anchor location, and will become a candidate if it's moving towards the anchor location. Next, each candidate node will calculate their individual backoff time before sending the acknowledgement (ACK). The backoff time $T_{backoff}$ is defined by:

$$T_{backoff} = \frac{(d - d_0 + r)}{2r} T_{max}, \tag{1}$$

where $T_{max}$ is the maximum backoff time set by the system; $d$ is the distance between the candidate node and the anchor location; $d_0$ is the distance between the carrier node and the anchor location; $r$ is the radio range. Using this equation, we can distribute the ACK backoff times between $[0, T_{max}]$. More importantly, the node with the shortest distance to the anchor location will have the smallest backoff time, and send out ACK the earliest, thus becoming the next carrier.

The new carrier will carry the Geocache until it finds out its distance to the anchor location starts increasing. It will then initiate another handoff to look for new carrier nodes using the same procedure.

**Trajectory-Based Selection:** While the distance-based approach works well for geographic routing in ad-hoc networks, it may not be suitable for vehicular networks because it ignores the fact that vehicles only move along fixed roadways. Therefore, progress in euclidian distances does not always yield a feasible path that returns to the anchor location. For instance, node $E$ in Fig. 3 is on a path (dead end) that never lead to $A$.

The above concerns lead us to the trajectory-based selection approaches. These approaches select new carrier nodes from those that are traveling in the opposite direction of the same trajectory passed by the Geocache. The rationale is that the trajectory describes a general feasible return path (with the exception of one-way paths scenarios). The heuristic we consider in this study is thus called *RevTraj* (**Rev**erse **Traj**ectory). Under this scheme, in the same example given in Fig. 3, node $C$ which is in the opposite direction of $B$'s trajectory will be chosen as the next carrier node.

The key component of RevTraj is trajectory recording: the aggregated path the previous carriers have traveled so far. The trajectory grows when a carrier is moving away from the anchor location, and shrinks when it's moving towards the anchor location. Depending on the storage and processing power available on the mobile units, we can use either raw GPS traces or "segmented" trajectory which only consists of the critical points on the path.

Next, let us look at the detailed handoff process in RevTraj. In the discussion below, we assume a segmented trajectory is used instead of a continuous trace. As illustrated in Fig. 4, a segment is represented by the coordinates of the two end points, e.g., $seg_1$: $[(-35941.32, 29577.19), (-35945.06, 29235.35)]$. The trajectory is implemented as a stack of end points, so that the latest segment is always on top of the stack. Below is the summary of the handoff procedure used in RevTraj:

1) *Handoff Initiation.* Non-first time Handoff occurs a divergence is detected from the recorded trajectory. The current carrier broadcasts the Geocache along with the trajectory.
2) *Candidate Identification.* Every node within the radio range pops out the latest segments from the trajectory stack. We use a parameter, *lookahead distance* (**LD**), to limit how many recent segments we exam-

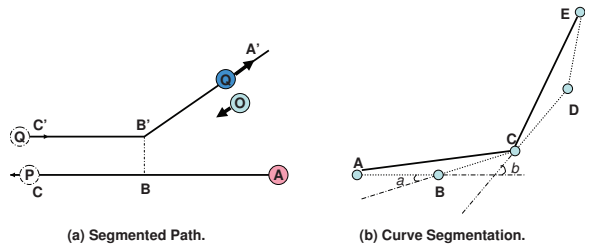



Fig. 5. Segmented path and curve segmentation.

ine. These lookahead segments can be numbered as $seg_1, seg_2, ...seg_{LD}$, with $seg_1$ being the latest segment. If the node finds itself on one of these lookahead segments, it becomes a candidate node and proceeds to the next step.

3) *Candidate Prioritization.* All the candidates are prioritized according to the following rules: (1) nodes traveling on higher-numbered segments are granted higher priority than those on lower-numbered segments; (2) for nodes traveling on the same segment, we give higher priority to those closer to the anchor location.
The prioritization rules can be easily implemented if each candidate node calculates its ACK backoff time using the following equation:

$$T_{backoff} = \frac{LD - i + \frac{d-d_0+r}{2r}}{LD} T_{max}, \quad (2)$$

where the definitions for $d$, $d_0$, $T_{max}$ are the same as before, and $i$ is the segment number.

4) *Carrier Selection.* The node with the smallest ACK backoff will send out the ACK earliest among all the candidates. To avoid hidden or exposed terminal problem, we suggest ACK be sent using higher transmission power to cover a wider range. Upon receiving the ACK, the old carrier as well as other candidates can decide whether to delete the Geocache, or keep it for some amount of time to increase the overall reliability..
5) *No Acknowledgement.* If the current carrier does not receive any ACK, it keeps the Geocache and initiates another handoff after a short interval.

We further distinguish between prioritized RevTraj as described above, and non-prioritized RevTraj, where all candidate nodes pick a random backoff value from $[0, T_{max}]$, and the node with the smallest backoff becomes the new carrier.

## IV. Trajectory construction in RevTraj

The primary challenge in implementing the trajectory-based boomerang protocol lies in the construction of the Geocache trajectory, and the detection of divergence from a given trajectory. To illustrate these challenges, let's consider the example in Fig. 5, where the Geocache was handed off to $Q$ who was traveling in the opposite direction of $P$'s trajectory. After traversing the path until $B'$, $Q$ diverges from $P$'s trajectory and heads for $A'$. In this case, $Q$ must be able to detect the divergence and start extending the path history from $B'$, so that at its next handoff, a modified path $A \rightarrow B(B') \rightarrow A'$ will be passed to the new carrier node $O$. The challenge in this implementation lies in developing robust trajectory update and divergence detection algorithms that are feasible across a variety of road networks.

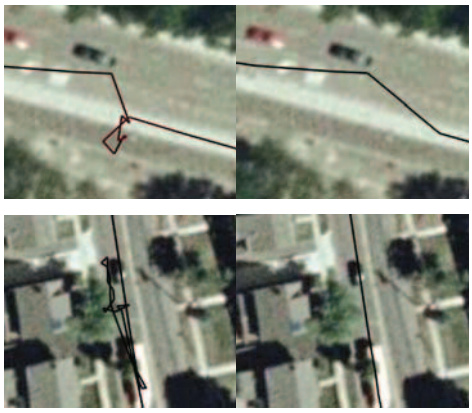


Fig. 6. Traces before (two pictures on the left) and after (two pictures on the right) data pre-processing.

### A. Data Pre-processing and Trajectory Recording:

In RevTraj, we need to construct trajectories from location (latitude and longitude) recordings reported by the GPS. First, we aggregate consecutive samples with little spatial distance in between (20m in our experiments), to reduce sample noise. The effect of this pre-processing is illustrated in Fig. 6.

Next, we segmentize the path, retaining only critical turning points by comparing the heading difference between the node's driving direction and the direction of the current segment. If the heading difference exceeds a threshold, we decide we are heading at a new direction and add the turning point to the trajectory to mark the start of a new segment. Consider the consecutive GPS recordings $A, B, C, D, E$ as illustrated in Fig. 5(b). Initially the heading is the direction of $AB$. When $C$ is present, we check the angle $\alpha$ between $BC$ and $AB$, and determines $\alpha$ is small enough to consider $C$ still on the same segment as $AB$. Next when $D$ is recorded, we check the angle $b$ between $CD$ and $AB$. At this time, $b$ is above the threshold, therefore we identify a new segment $CD$. At the end of this example, we have two segments: $AC$ and $CE$.

In the implementation, we also defined a threshold for the minimum segment length, to deal with large curve scenarios with consecutive small angle differences.

### B. Divergence Detection:

When on the return path to the anchor location, a node shrinks the saved trajectory by removing segments it has passed. Meanwhile, it also needs to continuously check if it has diverged from the remaining trajectory.

Intuitively, a divergence from the trajectory will result in a noticeable change in the heading direction, as well as a

Fig. 7. The routes we traversed in the experiment is colored in red.

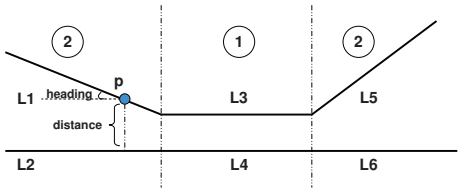



Fig. 8. Divide route pairs into 2 groups.

distance increase from the trajectory. However, using one factor alone to determine divergence could be erroneous. Lane shift, the individual's driving behavior and many other factors may all lead to a sudden direction change without actual divergence. Further, the variance in road widths (e.g., 15 to 60 ft for city roads[2]) makes the selection of a single distance threshold difficult.

In our divergence detection algorithm, we monitor the following conditions when a new GPS data is generated: (1) if the distance $d$ between the current location and the trajectory has exceeded the distance threshold $d_0$, and the heading change has exceeded the heading threshold $h_0$, (2) if $d$ has exceeded the maximum road width $d_{max}$. Divergence is declared if either condition is met for $k$ consecutive GPS readings. Therefore, the rule for divergence detection is defined as:

$$divergence = \begin{cases} 1 & d > d_{max}, \text{ or} \\ & d > d_0 \text{ and } h > h_0; \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Here, $d_{max}$ is the maximum road width, which can be obtained from road design manuals. $d_0$ and $h_0$ are the thresholds for distance and heading difference. Next, we will discuss how these threshold values are determined based on analysis with a real-time traffic trace collected from the southern New Jersey area.

### C. Divergence Detection Model based on Real-world Traces

Given the divergence detection method in Eq. 3, the key to the solution is thus to set suitable values for the two thresholds:

[2]http://www.greensboro-nc.gov/visitors/

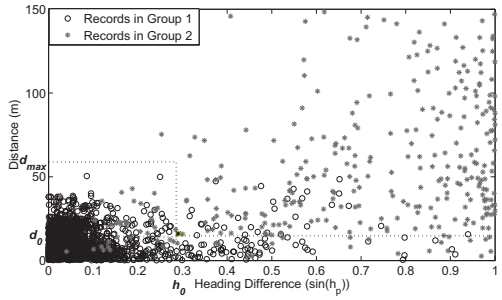



Fig. 9. The records of the two groups.

$d_0$ and $h_0$. In this study, we use the GPS samples taken from a field study to determine appropriate threshold values.

We collected 2 hours of GPS traces in the New Brunswick and North Brunswick area in New Jersey. We drove on both highways and local streets, covering about 55 miles with an average speed of 35 Mph. Fig. 7 depicts out route on the local map. In the experiment, we covered the loop shown in Fig. 7 twice. In the first pass we strictly stayed on the main loop, while in the second pass, we constantly drove into detour and side streets to emulate divergence from the main loop.

Next, we overlay the traces from the two passes and manually divide them into segment pairs. As shown in Fig. 8, in each segment pair the two paths either diverge or remain parallel. The segment pairs are then manually labeled into 2 groups: Group 1 for parallel pairs (e.g. the $\{L3, L4\}$ pair in Fig. 8) or Group 2 for diverging pairs (e.g. the $\{L1, L2\}$ pair in Fig. 8).

We then process the segment pairs in each group as the following. Taking the $\{L1, L2\}$ pair for example, for each location record $p$ on $L1$, we create a new record ($d_p$, $h_p$), where $d_p$ is the distance from $p$ to $L2$, and $h_p$ is the heading difference between $p$ and $L2$. Fig. 9 plots the distance and heading difference values of all records from both groups (5444 records in total). We apply a heuristic based classification method, and with $d_0 = 16$m and $h_0 = 0.29$, we are able to achieve a detection rate of 98.7%. The average detection delay is 2.26(sample), meaning the divergence will be accurately detected at the 3rd sample from the start of the divergence. $d_{max}$ is set to 60m according to the street design manual for the city of New Brunswick, NJ.

## V. Return Probability Analysis in Manhattan Grid

In this section, we analyze the performance of Geocache anchoring protocols in terms of return probability using a Manhattan grid topology. The return probability represents the likelihood for the protocol to return the Geocache back to the anchor location.

To simplify the analysis, we make the following assumptions: (i) the nodes are uniformly distributed on the roadways, (ii) grid blocks are of the same unit-size length, (iii) the radio range for all nodes are the same, which is also the unit-size, so that at any time only one intersection is under the radio range's

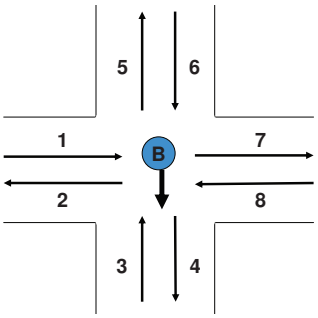



Fig. 10. 8 possible directions for a car at a crossing.

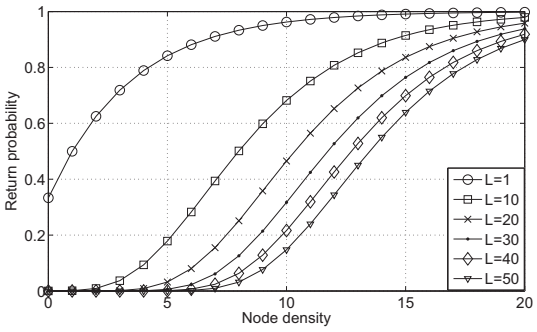



Fig. 11. Return probability for prioritized RevTraj with varying distance and node density.

coverage, (iv) the probability for a node to turn left, right, or move straight at an intersection is equally set to $p_t = \frac{1}{3}$ (assuming no U-turns), and (v) in system implementation, if the first handoff is not successful (due to low node density, etc), we allow further handoff attempts after a certain time interval. However, to simplify the analysis, for both protocols we only consider a single attempt for each handoff.

We note that we use the above assumptions to simplify the analysis, and we can apply the results derived from the analysis to other situations as well.

### A. Return Probability Analysis for RevTraj

In this subsection, we discuss the Geocache return probability if carriers are chosen based on a recorded trajectory such as in RevTraj.

First, we calculate the probability that a suitable carrier node is available (within the transmission range) when handoff occurs. Due to assumption (iv), the probability for one node at a four-way intersection to follow a fixed trajectory is $\frac{1}{4}$ (as shown in Fig. 10, the probability that the node is already on the trajectory with the correct direction, which is $\frac{1}{8}$, plus the probability that the node will turn to the trajectory, which is $\frac{3}{8} \times \frac{1}{3}$). Therefore, the probability to find at least one node on the trajectory given $d$ nodes available in radio range is $p_h = 1 - (1 - \frac{1}{4})^d$.

Next, we calculate the return probability for a complete trajectory with length $L$. By length $L$, we mean there are $L$ remaining segments on the trajectory other than the segment the node is currently on. Therefore, for a path with length $L$, there are $L$ remaining intersections to the anchor location. At each intersection, the node either follows the trajectory with probability $p_t$, or needs to hand off with probability 1-$p_t$. Therefore, in RevTraj, the return probability $P_{return}$ with path length $L$ is defined as:

$$P_{return} = [p_t + (1 - p_t)p_h]^L. \quad (4)$$

Fig. 11 plots the return probability for prioritized RevTraj. We vary the distance $L$ from 1 to 50, and the results show that the return probably increases when the node density increases, and gradually approaches 1 as the node density reaches a certain level.

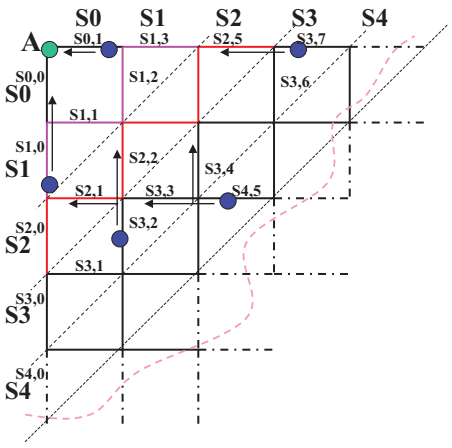



Fig. 12. Dividing grid into groups of segments.

### B. Return Probability Analysis for MaxProgress

In this subsection, we discuss the Geocache return probability if carriers are chosen according to its distance to the anchor location such as in MaxProgress.

We start by labeling the road segments on a grid. Fig. 12 depicts a fully-connected grid with $A$ as the anchor location. First, we divide the segments into different sets $S_i$ based on their distance to the anchor location. In Fig. 12, we are showing 5 sets, $S_0$ to $S_4$. The $i$th set $S_i$ contains $2i + 2$ segments. We label each segment as $s_{i,j}$, where $i$ is its set number, and $j$ ($0 \leq j \leq 2i+1$) is the segment's index number within set $S_i$. In Fig. 12, the segments within the same set are numbered from lower left to upper right.

Next, we compute the return probability $P_{i,j}$ for segment $s_{i,j}$, which is the Geocache's return probability when it's currently located on segment $s_{i,j}$. We distinguish four classes of segments: (1) segments adjacent to $A$ (in $S_0$), (2) segments on or adjacent to the left vertical edges where $j = 0$ or $j = 1$, (3) segments on or adjacent to the upper horizontal edges where $j = 2i$ or $j = 2i + 1$, and (4) all remaining segments.

According to the protocol, handoff will occur as soon as the distance between the carrier and the anchor location starts to increase, suggesting the carrier diverges from the shortest path to the anchor location. As shown in Fig. 12, for the two

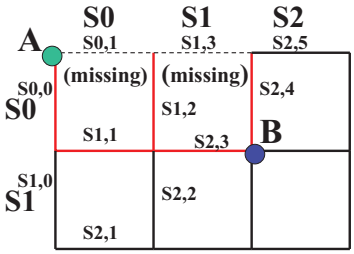



Fig. 13. An example of partially-connected Grid.

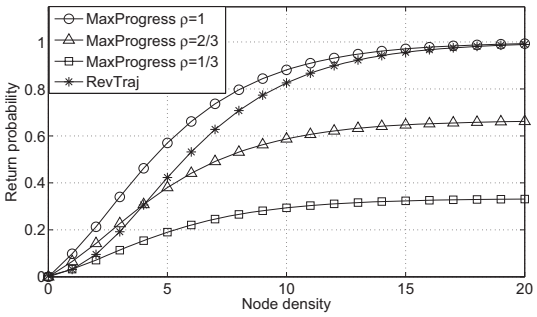



Fig. 14. Comparing MaxProgress and RevTraj in grids with different connectivity level.

segments connecting the anchor location in case (1), their return probability is 1. In cases (2) and (3), there is only one choice to remain on the shortest path (e.g., for $s_{2,0}$, the immediate next segment along the shortest path to $A$ is via $s_{1,0}$). The node may turn into the shortest paths itself, or find a node that is or will be on the shortest path with probability $k_1 = 1 - (\frac{3}{4})^d$. In case (4), for each segment, there are two choices of shortest path, so similarly, the probability $k_2$ for the Geocache to remain on the shortest path after the intersection is $k_2 = 1-(\frac{1}{2})^d$. Therefore, we give our recursive equation for the return probability in Eq. 5. $P_{i,j}$ equals to the probability for the Geocache to get onto the shortest path(s), times the return probability of the chosen shortest path(s).

$$P_{i,j} = \begin{cases} 1 & \text{for } i = 0; \\ (\frac{1}{3} + \frac{2}{3}k_1)P_{i-1,0} & \text{for } j \in [0,1]; \\ (\frac{1}{3} + \frac{2}{3}k_1)P_{i-1,2i-1} & \text{for } j \in [2i, 2i+1]; \\ (\frac{1}{3} + \frac{1}{6}k_2)(P_{i-1,j-2} + P_{i-1,j-1}) & \text{otherwise,} \\ & \text{when } j\text{'s odd;} \\ (\frac{1}{3} + \frac{1}{6}k_2)(P_{i-1,j} + P_{i-1,j-1}) & \text{otherwise,} \\ & \text{when } j\text{'s even.} \end{cases} \quad (5)$$

### C. Return Probability Analysis in Partially-connected Grids

After analyzing the return probability in a fully-connected grid, we next look at the return probability for MaxProgress in partially-connected situations, which are more common in real life.

First, we define a *successful path* as a shortest path that can lead to the anchor location from the hand off location in a partially-connected grid (e.g., the path$\{s_{2,3}, s_{1,1}, s_{0,0}\}$ from $B$ to $A$ in Fig.13). We can always generate a partially-connected grid with smaller $\rho$ by removing segments from a fully-connected grid. For instance, the incomplete grid shown in Fig. 13 is generated by removing segments ($s_{0,1}$ and $s_{1,3}$) from a full grid, and the value of $\rho$ according to Eq. 6 is $\frac{1}{3}$.

Next, we introduce a parameter

$$\rho = \frac{N_{success}}{N_{total}}, \quad (6)$$

where $N_{success}$ is the number of successful paths in the partially-connected grid, and $N_{total}$ is the total number of paths that head to (but not necessarily reach) the anchor location's direction. For a fully connected grid, like the one in Fig. 12, we have $\rho = 1$.

To apply Eq. 5 on the example shown in Fig. 13, we assign 0 to $P_{0,1}$ and $P_{1,3}$ to invalidate the two missing segments. Therefore, assuming $P_t = \frac{1}{3}$, with $A$ as anchor location and $B$ as handoff location, we have the return probability in the partially-connected grid as:

$$P = (\frac{1}{3} + \frac{1}{6}k_2)(\frac{1}{3} + \frac{1}{6}k_2)(\frac{1}{3} + \frac{2}{3}k_1), \quad (7)$$

while the return probability from B to A in an otherwise fully-connected grid is:

$$P_{complete} = (\frac{1}{3} + \frac{1}{6}k_2)(\frac{1}{3} + \frac{2}{3}k_1)(1 + \frac{2}{3}k_1 + \frac{1}{3}k_2). \quad (8)$$

Therefore we have

$$\frac{P}{P_{complete}} = \frac{\frac{1}{3} + \frac{1}{6}k_2}{1 + \frac{2}{3}k_1 + \frac{1}{3}k2}. \quad (9)$$

The equation $\frac{P}{P_{complete}}$ achieves its maximum $\frac{1}{3}$ when both $k_1$ and $k_2$ are equal to 0. That is, $\frac{P}{P_{complete}} \leq \frac{1}{3}$. Note that for this particular topology, $\rho$ also equals to $\frac{1}{3}$. In fact, we have examined numerous grid-based road topologies and we are always able to observe $\frac{P}{P_{complete}} \leq \rho$ for the cases we have studied.

This is a powerful observation, and it also confirms our hypothesis about distance-based approaches such as MaxProgress: the return probability will degrade when the connectivity of the road network decreases. However, in the trajectory-based approach, the return probability actually increases when removing segments from the fully-connected grid according to Eq.4. In the real world, the road topology approximating a fully-connected grid is very rare, but rather most of the road maps have medium to low $\rho$ values, which may severely degrade the return probability using MaxProgress.

Fig. 14 compares the return probability of RevTraj with MaxProgress using Eq. 4 and Eq. 5. $L$ is fixed to $L = 5$. For MaxProgress, we show the Geocache return probability with varying $\rho$ value. For RevTraj, we show its minimum return probability value when $\rho = 1$. We observe that for MaxProgress, the return probability degrades significantly when $\rho$ decreases, while the RevTraj outperforms MaxProgress with medium and low $\rho$ values, which corresponds to typical road topologies in the real world.

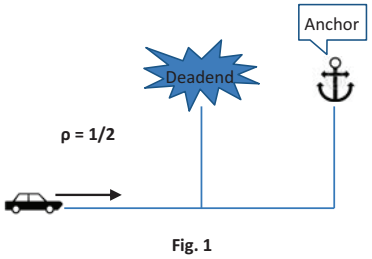
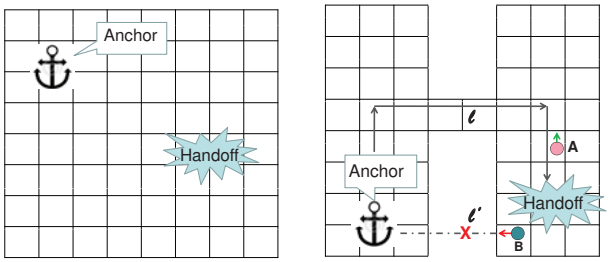
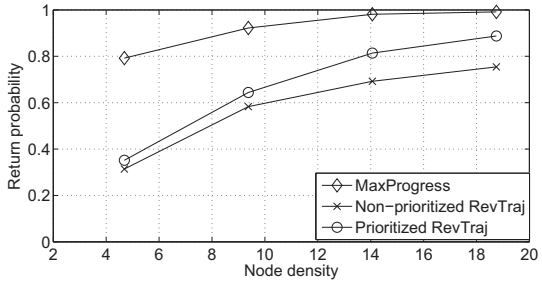



Fig. 15. Illustration of deadend's impact to $\rho$ in a roadmap.

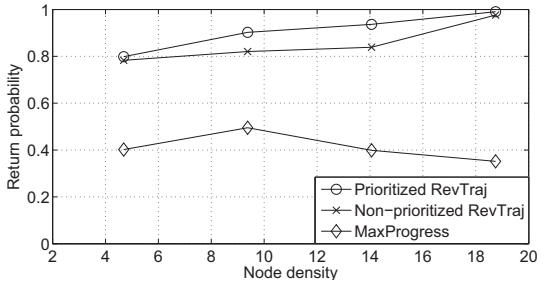



(a). Fully-connected grid. (b). Two-city grid.

Fig. 16. Two road maps with different $\rho$ values. (a). A fully-connected grid with $\rho = 1$ representing the Manhattan city road map. (b). A partially-connected twin-city grid with low $\rho$ value, representing two cities being connected by a major highway.




(a). Geocache return probability in fully-connected grid.




(b). Geocache return probability in twin-city grid.

Fig. 17. Comparing RevTraj with MaxProgress using two road map settings. MaxProgress fares better with $\rho = 1$, but is significantly outperformed by RevTraj under a low $\rho$ value.

## VI. Performance Evaluation

In this section, we study the performance of Geocache anchoring protocol through simulation. We measure the return probability of the Geocache when varying the vehicular density and the connectivity of the road map.

### A. Effect of Road map Connectivity

As we have discussed in the previous section, intuitively, the distance-based MaxProgress works better under fully or mostly connected road map topologies. We capture the connectivity characteristic using parameter $\rho$ given in Eq. 6. Our definition of $\rho$ is different from the general concept of road connectivity. Recall that in our definition, $\rho$ at a location is the ratio between the number of paths that pass this location and can reach the anchor location and the total number of paths that pass this location. For both the numerator and the denominator, we only consider shortest-distance paths, i.e. we do not backtrack once the map hits a dead end. According to this definition, road conditions such as dead ends will result in a relatively low $\rho$ value in the roadmap. Fig. 15 illustrates this point.

A well-connected city road map such as Manhattan's is a good example for road topologies with large $\rho$ value, where all road segments have outlets at both ends. Many other areas, however, do not have this property because dead ends are common in either urban or rural road systems. According to [10], in the Digital Road Map Data Base (DRMap) for Japan, among all the 354,000 investigated roads, there are 22,000 dead ends in total. In partially-connected areas, we expect MaxProgress to exhibit suboptimal performance. An example is given in Fig. 16(b). The Geocache was handed off after traveling along a trajectory $l$. When choosing the next carrier, MaxProgress always favors those that are physically closer to the anchor location, which is $B$ in this case. But the nature of the road map makes this choice a bad decision, because the seemingly shortest path $l'$ does not exist in the actual road map due to low connectivity, whereas the longer alternative $l$ is the only feasible path leading back to the anchor location. On the other hand, according to RevTraj, node $A$ who's following the trajectory $l$ will be chosen as the next carrier. By following the trajectory, no matter how poorly connected the road map is, we are always confident that the trajectory can lead to the anchor location. A trade-off is the probability for RevTraj to find a carrier is lower than that of MaxProgress. Especially with low node density, RevTraj may suffer from not being able to find a carrier that's exactly on the trajectory.

To verify the above hypothesis, we simulated the performance for the two protocols on the two road maps depicted in Fig. 16(a) and (b). Fig. 16(a) represents a well-connected city road map with $\rho = 1$, while Fig. 16(b) shows a road map with a low $\rho$ value, representing two cities being connected by a major highway. The simulator of choice is *NS-2*, with 802.11 as MAC and PHY layer protocol. The radio range is 250m. At each intersection, the probability for a car to turn left, right and go straight is equally $\frac{1}{3}$. The length of each segment is 300m. The vehicular has constant speed of 30m/s. In RevTraj, the number of look ahead segments is 3, meaning we look at the most recent 3 segments when comparing the trajectory. Fig. 16 also indicates the approximate anchor location and handoff location for the two topologies.

Fig. 17 present the return probability of the two proto-

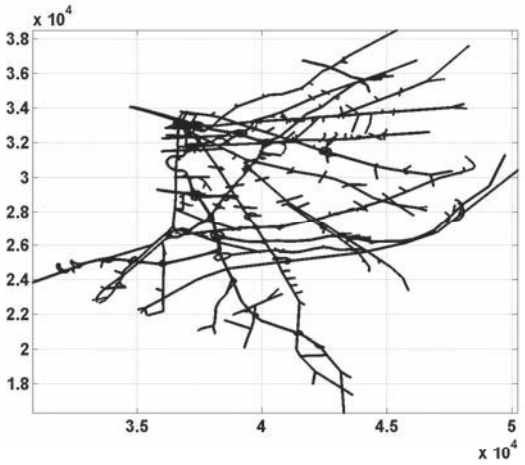



Fig. 18. We also evaluated the anchoring protocols using real traffic data from south New Jersey. This is the road map for the traffic data.

cols. As expected, in Fig. 17(a) which shows the results for the fully-connected grid topology, MaxProgress outperforms RevTraj. This is consistent with the analysis results shown in Fig. 14 where the return probability of MaxProgress is generally higher than that of RevTraj with $\rho = 1$. But in the partially-connected grid topology in Fig. 17(b), MaxProgress's return probability degrades severely, while RevTraj shows much better performance. This is again consistent with Fig. 14 where the return probability of MaxProgress with small $\rho$ is generally lower than that of RevTraj. Therefore, the results in Fig. 17 strongly confirm our hypothesis and analysis in Section V.

### B. Evaluating Anchoring Protocols

After studying the example scenarios, we next investigate the protocols' performances using a synthetic traffic trace collected from the PARAMICS model. The Paramics simulation model is based on southern New Jersey high way system and traffic topology (as shown in Fig. 18). It captures the interactions of real world road traffic through a series of complex algorithms that describe car following, lane changing, gap acceptance, and spatial collision detection. The trace contains $984,445$ records collected by $5000$ cars for a duration of $3395$ seconds during the 6am-7am off-peak traffic period. In our simulator, free-space propagation is used as communication model.

Every result we show in this section is averaged over $5000$ simulation runs. In each run, we select a random car's location at a random time as the anchor location, and let that car drive for a period of $T_h$ before handing off the Geocache. We end a simulation run either when a successful return is made or after $T_{end}$ elapses but still no successful return. After completing $5000$ simulation runs, the return probability is then calculated as the ratio of the number of successful returns over $5000$.

The return probability results are shown in Fig. 19. Since the node density is fixed in the trace, we vary the radio range in the experiments to change the number of cars covered in

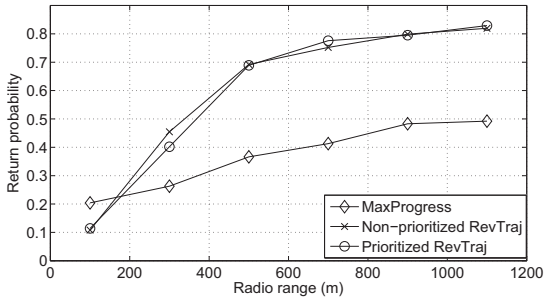
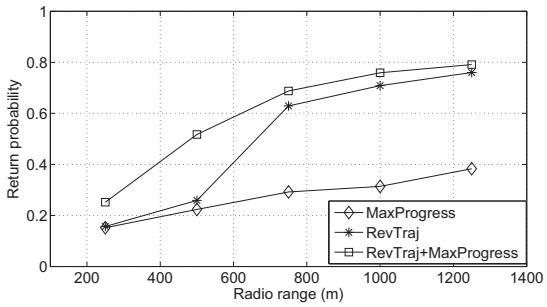
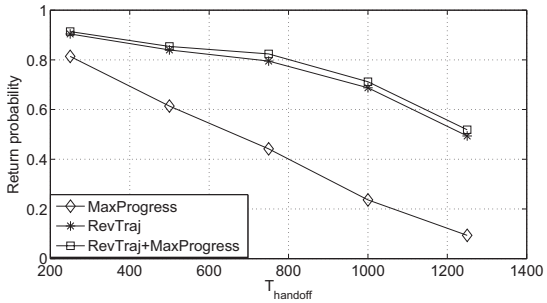



Fig. 19. The comparison of three anchoring schemes when increasing the radio range. $T_h = 750$ seconds.




(a). Return probability when varying radio range values with $T_h$ of 1000 seconds.




(b). Return probability when varying $T_h$ with radio range of 750 meters.

Fig. 20. Performance evaluation for the adaptive anchoring algorithm.

the handoff range, thereby simulating varying node densities. $T_h$ is set as $750$ seconds. We find that except for extremely short radio ranges (100m), corresponding to extremely low node density in our case, RevTraj significantly outperforms MaxProgress, with an improvement of about $70\%$. This suggests that many real-world road maps are of small $\rho$. Finally, we notice that all three schemes benefit from a larger radio range, which in our case, suggests that the anchoring protocols benefit from larger vehicular densities.

### C. Adaptive Anchoring Scheme

Through the previous simulation results, the two anchoring protocols exhibit their individual advantages under different circumstances. Therefore, it is natural to develop a protocol which combines the two and take advantage of both their strengths. In our enhanced scheme, we still keep the trajectory to ensure a guaranteed return path to the anchor location.

When we can not find suitable carriers via RevTraj, we extend our search by including nodes found using MaxProgress. At all time we keep the trajectory, hoping that later carrier nodes may return back to the trajectory, or at least follow the general direction.

In Fig. 20(a), we set $T_h = 1000$ seconds and vary the radio range of the mobile nodes. Results show that across all radio ranges we picked, the adaptive protocol always returns Geocache with the highest probability. RevTraj exhibits lower return probability when node density is low since it misses many opportunities to conduct a successful return if MaxProgress was used. MaxProgress shows the worst performance among the three. In Fig. 20(b), we compared the three protocols with constant radio range of 750m while varying $T_h$. We observe the adaptive approach still achieves the highest return probability. And for all the three protocols, longer hand off time leads to lower return probabilities.

## VII. Adaptive Handoff for In-time Anchoring

After studying how to return Geocache to the anchor location in the general setting, we next look at a specific anchoring requirement, *in-time anchoring*. Here, the Geocache is required to return to the anchor location within a specific time interval. A wide range of mobile applications have such requirements. For example, a mobile user may have a query "what is the average car speed around my location for the next 5 minutes?" He only accepts real-time responses within the next few minutes, and after that, he will lose interest and leave.

The challenge here is two-fold. First, we would like to have in-time returns. Second, we would like the return time to be as close to the expected return time as possible. If the Geocache is returned too early, we need to either keep the Geocache exactly at the anchor location until the deadline, which incurs high communication overhead, or continue to boomerang the Geocache, with the risk of late returns.

To meet this challenge, we need to carefully control the initial handoff time $T_h$ to ensure the Geocache's timely return. In this section, we discuss two such control algorithms.

### A. Addictive Adjust Multiplicative Decrease Handoff Policy

The first protocol is called **A**ddictive **A**djust **M**ultiplicative **D**ecrease (**AAMD**). Basically, we control $T_h$ based on the observed Geocache return time $T_r$. Intuitively, $T_h$ should be increased when the previous $T_r$ is well below the expected return time $T_{ERT}$, and should be decreased when the previous $T_r$ approaches or even exceeds $T_{ERT}$. We introduce a threshold rate $\beta (0 < \beta < 1)$, and we increase $T_h$ by $\Delta$ if $T_r < \beta T_{ERT}$, and decrease $T_h$ by $\Delta$ if $T_r \geq \beta T_{ERT}$. Here $\Delta$ is usually a small value compared to $T_{ERT}$, e.g., $\Delta = \frac{T_{ERT}}{20}$.

When we have a late return ($T_r > T_{ERT}$), $T_h$ is reduced by half in order to quickly get back to in-time returns. Therefore, the complete policy for AAMD is:

$$T_h = \begin{cases} T_h + \Delta, & \text{if } 0 < T_r < \beta T_{ERT} \\ T_h - \Delta, & \text{if } \beta T_{ERT} \leq T_r \leq T_{ERT} \\ \frac{T_h}{2}, & \text{if } T_r > T_{ERT} \end{cases} \quad (10)$$

### B. Q-Handoff: Q-Learning-based Handoff Policy

AAMD has two problems. First, it is very conservative by cutting $T_h$ by half when late return occurs, which will lead to a slow start for $T_h$ and therefore excessive communication overhead. Second, it does not take history information into consideration. We address these shortcomings in our second policy. Here, we first build an MDP (Markov decision process) model for the handoff adjustment scenario, then train the policy using a reinforcement learning approach: Q-learning. We also introduce an adaptive algorithm to adjust the *setPoint* which is used to reset $T_h$ in case of late returns. We call this handoff policy *Q-Handoff*.

*1) MDP Model for Q-Handoff:* Q-learning learns the expected utility of taking a given action in a given state, whose quality is represented by the function: $Q(s, a)$. Formally, the basic reinforcement learning model, as applied to MDPs, consists of:

- a set of environmental states $S$;
- a set of possible actions $A$;
- a set of scalar rewards $R$.

Therefore, we give the MDP model according to our problem scope:

- $S = (s_0, s_1, s_2, s_3, s_4, s_5)$;
  These states represent different Geocache return situations. $s_0$ represents late returns with $T_r > T_{ERT}$, while states $s_i$ (i$\neq$0) represent different in-time return situations where $\frac{(i-1)T_{ERT}}{5} < T_r \leq \frac{iT_{ERT}}{5}$. For example, if we have $T_r = 0.35T_{ERT}$, then the state is $s_2$.
- $A = (a_0, a_1, a_2, a_3)$;
  These actions define how to adjust the value of $T_h$. Specifically,
  - $a_0$: reset $T_h$;
  - $a_1$: increase $T_h$ by $\Delta$;
  - $a_2$: keep $T_h$ unchanged;
  - $a_3$: decrease $T_h$ by $\Delta$.

Next we define the reward function $r$. Between in-time returns and late returns, the reward function would favor the former. Further, among all the in-time returns, those that are closer to the expected return time are preferred. Therefore, we define the following reward function:

$$r = \bullet_{T_r < T_{ERT}} - \frac{|T_r - T_{ERT}|}{T_{ERT}}, \quad (11)$$

let $\bullet_{T_r < T_{ERT}}$ be the indicator that takes value 1 if $T_r < T_{ERT}$ and 0 otherwise.

We use the quality function $Q$

$$Q : S \times A \to \mathbb{R}$$

to calculate the score of an action in a certain state. At the beginning, $Q(s_i, a_j)$ is initialized with random values, and then gets updated iteratively by using:

$$Q(s_i, a_j) = (1 - \alpha) Q(s_i, a_j) + \alpha [r + \gamma \max_{a \in A} Q(s', a)], \quad (12)$$

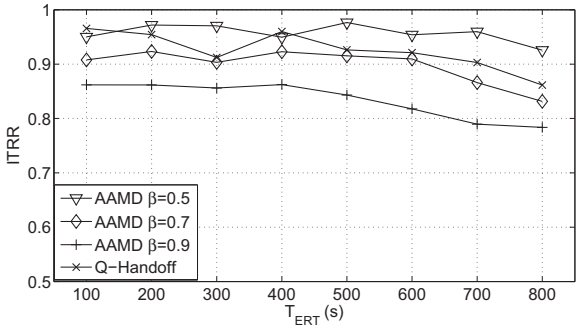



Fig. 21. In-time return probability for AAMD with various $\beta$ values and Q-Handoff Policy.

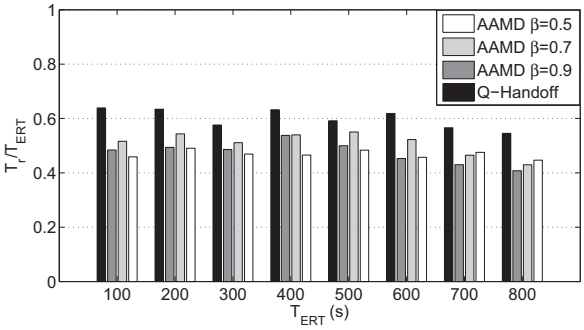



Fig. 22. Return time ratio for AAMD with various $\beta$ values and Q-Handoff Policy.

where $r$ is the immediate reward defined by Eq. 11, and $s'$ is the next state. The first term evaluates the importance of the history information, where the learning rate $\alpha(0 < \alpha \leq 1)$ determines to what extent the recent information will override the old information. The second term in the equation evaluates the immediate reward and the expected future reward, where $\gamma(0 < \gamma \leq 1)$ is the discount factor.

Therefore, we have our $\epsilon$-greedy Q-Handoff policy:

**Policy:** At each state $s_i$, the agent picks the action $a = \text{argmax}_a\, Q(s_i, a)$ with probability 1-$\epsilon$, and picks a random action with probability $\epsilon$.

Using this policy, we guarantee that most of the time we act greedily, selecting actions that lead to the greatest reward. Occasionally, we select random actions to explore unknown space which may potentially lead to even higher rewards. In machine learning, $\epsilon$-greedy policies are commonly used to deal with the exploit-explore dilemma.

*2) SetPoint Adjustment:* In Q-Handoff, when a late return occurs, $T_h$ is reset to a *setPoint* value (action $a_0$). A ideal *setPoint* should have "good record", meaning previous $T_h$ values around the *setPoint* have resulted in a large number of successful in-time returns. A practical approach here is to save all the successful $T_h$ values but give more weight to recent ones, since recent $T_h$, especially those a few iterations before a reset usually yield in-time returns closer to the expected return time. Therefore, we use an Exponentially Weighted Moving Average (EWMA) to update the *setPoint* as the following:

$$setPoint = \theta T_h + (1 - \theta) setPoint. \tag{13}$$

### C. Performance Evaluation

We evaluate the two policies using NS-2, with 802.11 as MAC and PHY layer protocol. We use the same southern New Jersey traffic trace as in Section VI, with $T_{ERT}$ ranging from 100s to 800s. For each $T_{ERT}$, we run the simulation for $\frac{T_{ERT}}{10}$ times. For example, with $T_{ERT} = 100$s, we run the trace for 10 times. This is because for the constant-sized trace file (which covers approximately 5000s), large $T_{ERT}$ values will lead to fewer simulation results (number of results $\approx \frac{5000}{T_{ERT}}$). By varying the number of runs for different $T_{ERT}$, we can guarantee sufficient simulation results for each $T_{ERT}$ value.

In AAMD, we choose $\beta$ from $(0.5, 0.7, 0.9)$. In the $\epsilon$-greedy Q-Handoff policy, we set the learning rate $\alpha$ as 0.1, the smoothing factor $\theta$ for EWMA as 0.1, and $\epsilon$ as 0.01. For both protocols, the addictive adjustment amount is $\Delta = \frac{T_{ERT}}{20}$.

The first metric we look at is the in-time return rate (ITRR). Fig. 21 shows the ITRR for Q-Handoff and AAMD with different $\beta$ values. For AAMD, smaller $\beta$ values yield earlier returns, thus higher ITRR, and the average ITRR ranges from $83.4\%(\beta = 0.9)$ to $95.73\%(\beta = 0.5)$. Using Q-Handoff, we can achieve an average ITRR of $92.5\%$.

Next we compare the average return time of the different policies. The metric we use here is the ratio of the average return time to the expected return time. It is an important metric because it reflects an algorithm's ability to adjust the handoff time to meet certain return time constraints. Further, small return time usually indicates frequent handoff, and thus higher communication overhead. Fig. 22 shows that Q-Handoff yields closer-to-expectation returns. Its average ratio is $0.60$, which is $20\%$ better than the best ratio of AAMD when $\beta = 0.7$.

## VIII. Related Work

This work spans the fields of mobile sensor networks and vehicular networks. Perhaps closest in spirit to the Geocache programming abstraction are geographic hash tables [11], which provide a programming interface for data-centric storage in stationary sensor networks. Spatialviews [12] provides location-oriented programming language abstractions for mobile ad hoc networks, to ease application development and maintenance. This work does not address distribution of information at the protocol level, which is a key focus of this paper.

*a) Mobile sensor networks:* Recent works in mobile sensor networks exploit mobility when it is not feasible to build a dense network of fixed sensors. Notably, Zebranet [1] places sensors on zebras to collect valuable zoology data. In under water sensor network [13], mobile nodes are robots that collect data from regions of interest. Several projects target specifically at vehicular sensing. CarTel [2], for example, is a comprehensive distributed mobile computing system used to collect, process and visualize data from sensors located

on mobile units. It aims at exploring in-network computing on individual mobile units, as we do, but it does not use inter-vehicle communication, which in our project, is a main focus to enable distributed aggregation of sensor readings from multiple cars. Another vehicular sensor network: MobEyes [3], [14], introduces MDHP (MobEyes Diffusion/Harvesting Processor), a protocol used to spread information within wireless sensor networks and build low-cost index of mobile storage. Although our projects bear similarities in that we also aim to develop low-cost yet efficient inter-vehicle communication protocol, MobEyes relies largely on an opportunistically broadcast approach, possibly with the emphasis of simplified protocol, while we aim at minimizing traffic overhead caused by more sophisticated schemes.

*b) Inter-vehicle, geographic, and delay-tolerant communication:* Many projects have addressed scalable communication in mobile ad hoc networks (e.g., [6]), in sparse or disconnected mobile ad hoc networks (e.g., [15]–[18]), or through Infostations. In [19], the authors introduce Infostations to deliver data to mobile nodes. In [20], [21], the authors aim at providing location-specific information to mobile devices, in which they developed schemes for detecting and transferring information of interest. All of these techniques adopt a server-client approach, but in our case, the information is provided by mobiles that have passed the location. The MaxProp [16] routing protocol is used to ensure effective routing of DTN (disruption-tolerant networks) messages via intermittently connected nodes. These protocols are based on different communication workloads, such as unicast between randomly chosen nodes, or multicast to random node sets. These techniques focus on delivering messages to certain nodes, while our protocols try to keep information around a certain location. In [15], designated mobile nodes (message ferries) store and carry messages. Our project differs in that virtually all nodes are "peers". In [17], the authors aim to guarantee message transmission in minimal time, at the expense of additional messaging overhead. Instead, our applications are more delay tolerant, and the main goal is to reduce communication overhead.

Geocast protocols [7], [22], [23] transmit messages to a predefined geographical region. They are suitable for location-based services such as position-based advertising and publish-and-subscribe. Repeated geocasts or time stable geocasts [8] could also be used to maintain Geocache in a certain area and bear similarities to our baseline scheme. It is different in concept though in that it requires the definition of a geographic region, which is not needed in our case. Most geocast schemes concentrate on routing messages to the areas of interest, or distributing messages to all nodes [7], [23], while Geocache is established close to the anchor location and needs only be known to very few nodes. Further, time-stable geocasts continuously remain in the region of interest, while Geocache can travel away from the anchor location.

In [24], it mentions some trajectory concepts, but it fails to take into account the peculiarities of vehicular networks and still only forwards data to a node that is physically closer to the destination. Geopps [25], [26] are maybe the most similar works to ours, however, it requires each mobile node to have full topology information which is not feasible in realistic scenario.

In [27], the authors examine the dissemination of availability reports about resources in mobile peer-to-peer networks. By opportunistically propagating exchanging the reports, and decaying the relevance of the report as its age increases, the proposed algorithm is able to limit the global distribution of a report to a bounded spatial area and to the duration for which it is of interest. Although we are also interested in retaining spatio-temporal information to a local area, our focus is to maximize the probability to find certain information, instead of bounding its global distribution to a certain spatial area and duration.

*c) Matching GPS observations:* The problem of map matching based on GPS readings have been extensively studied. Some existing work include [28] [29] [30] [31]. Even though we share some similarities with the map matching problem when using gps readings to identify road segments, we differ significantly with the general map matching problem in the use of road maps. Map matching solutions generally focus on matching a node's position to the nearest street presented in the map. This differs fundamentally from our work since we don't use street maps but only GPS readings of traversed paths. Therefore, the general map matching approach which involves searching and comparing nearby road segments could not be applied to our problem. Instead, we propose using absolute distances and heading differences with the recorded road segment to determine divergence.

## IX. Conclusions

We have presented the trajectory-based boomerang protocol to periodically make available data at certain geographic locations in a highly mobile vehicular network. The boomerang protocol returns the Geocache through nodes traveling toward the anchor location. To increase the probability of successful return, it records a node's trajectory while moving away from the anchor location then select nodes to return the Geocache based on the trajectory (RevTraj). We compared this scheme with a shortest-distance georouting scheme MaxProgress, and demonstrated that our scheme significantly outperforms its counterpart in realistic traffic simulation, with a return probability improvement of up to $70\%$. We also extend the boomerang protocol to satisfy more stringent anchoring requirements, such as returning the Geocache within specified time limits. This is achieved through adapting the initial handoff time based on the return time history.

## References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," *ACM SIGOPS Operating Systems Review*, vol. 8, pp. 96–107, 2002.

[2] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: a distributed mobile sensor computing system," in *Proc. of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 125–138.

[3] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "Mobeyes: Smart mobs for urban monitoring with a vehicular sensor network," *Wireless Communications, IEEE*, vol. 13, pp. 52–57, 2006.

[4] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J. Herrera, A. Bayen, and Q. J. M. Annavaram, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *Proc. of the 6th International conference on Mobile Systems, Applications, and Services*, 2008, pp. 15–29.

[5] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. of the 6th International conference on Mobile Systems, Applications, and Services*, 2008, pp. 29–39.

[6] J. Li, J. Jannotti, D. Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proc. of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 120–130.

[7] Y. B. Ko and N. H. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 7, pp. 471–480, 2002.

[8] C. Maihofer, T. Leinmller, and E. Schoch, "Abiding geocast: Time-stable geocast for ad hoc networks," in *Proc. of the 2nd ACM international workshop on Vehicular ad hoc networks*, 2005, pp. 20–29.

[9] M. Mauve and J. Widmer, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, pp. 30–39, 2001.

[10] K. O. S. Kamijo and A. Kitamura, "Orbit radio grid tested for evaluation of next-generation wireless network protocols," in *Proc. of the first vehicle and navigation and informaton systems conference*, 1989, pp. 308–309.

[11] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: A geographic hash table for datacentric storage," in *Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 78–87.

[12] Y. Ni, U. Kremer, A. Stere, and L. Iftode, "Programming ad-hoc networks of mobile and resource-constrained devices," in *Proc. of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, 2005, pp. 249–260.

[13] I. Vasilescu, K. Kotay, and D. Rus, "Data collection, storage, and retrieval with an underwater sensor network," in *Proc. of the 3rd International conference on Embedded networked sensor systems*, 2004, pp. 154–165.

[14] U. Lee, E. Magistretti, M. Gerla, P.Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 882–901, 2009.

[15] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proc. of the 5th ACM international symposium on Mobile ad hoc networking and computing*, 2004, pp. 187–198.

[16] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Routing for vehicle-based disruption-tolerant networks," in *Proc. of the 25th IEEE International Conference on Computer Communications*, 2006, pp. 1–11.

[17] Q. Li, D. Rus, M. Dunbabin, and P. Corke, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Proc. of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 44–55.

[18] L. Briesemeister and G. Hommel, "Role-based multicast in highly mobile but sparsely connected ad hoc networks," in *Proc. of the 1st ACM international symposium on Mobile ad hoc networking and computing*, 2000, pp. 45–50.

[19] R. H. Frenkiel, B. R. Badrinath, J. Borres, and R. D. Yates, "The infostations challenge: balancing cost and ubiquity indelivering wireless data," *Personal Communications, IEEE*, vol. 7, pp. 66–71, 2000.

[20] Y. cai and T. Xu, "Design, analysis, and implementation of a large-scale real-time location-based information sharing system," in *Proc. of the 6th International conference on Mobile Systems, Applications, and Services*, 2008, pp. 106–117.

[21] H. Lu, N. Lane, S. Eisenman, and A. Campbell, "Bubble-sensing: A new paradigm for binding a sensing task to the physical world using mobile phones," in *Proc. of the International Workshop on Mobile Device and Urban Sensing*, 2008.

[22] T. Small and Z. J. Haas, "The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)," in *Proc. of the 4th ACM international symposium on Mobile ad hoc networking and computing*, 2003, pp. 233–244.

[23] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto, "Carnet: A scalable ad hoc wireless network system," in *Proc. of the 9th ACM SIGOPS European Workshop*, 2000, pp. 61–65.

[24] J. LeBrun, C.-N. Chuah, D.Ghosal, and M.Zhang, "Knowledge-based opportunistic forwarding invehicular wireless ad hoc networks," in *Proc. of the 61st IEEE conference on Vehicular Technology*, 2005, pp. 2289–2293.

[25] I. Leontiadis and C. Mascolo, "Geopps: Geographical opportunistic routing for vehicular networks," in *Proc. of the IEEE Workshop on Autonomic and Opportunistic Communications*, 2007, pp. 1–6.

[26] I. Leontiadis and C. Mascolo, "Opportunistic spatio-temporal dissemination system for vehicular networks," in *Proc. of the 1st international MobiSys workshop on Mobile Opportunistic networking*, 2007, pp. 39–46.

[27] A.Sistla, O. Wolfson, and B. Xu, "Opportunistic data dissemination in mobile peer-to-peer networks," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, 2005, vol. 3633, pp. 923–923.

[28] C.E.White, D.Bernstein, and A.L.Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, pp. 91 – 108, 2000.

[29] C. S. Jensen and N. Tradisauskas, "Map matching," *Encyclopedia of Database Systems*, vol. 13, pp. 1692–1696, 2009.

[30] JS.Greenfeld, "Matching gps observations to locations on a digital map," in *Proc. of the 81th Annual Meeting of the Transportation Research Board*, 2002.

[31] M. Quddus, W. Ochieng, L. Zhao, and R. Noland, "A general map matching algorithm for transport telematics applications," *GPS Solutions*, vol. 7, pp. 157–167, 2003.

**Tingting Sun** is currently pursuing her Ph.D degree in Electrical and Computer Engineering at WINLAB, Rutgers University. Her Advisor is Prof. Yanyong Zhang, and she is also working with Prof. Wade Trappe and Prof. Marco Gruteser. Before joining Rutgers, she obtained her B.S. degree in the department of Computer Science at the University of Science and Technology of China. Her research interest includes location-aware systems, mobile networking, and resource management in WLAN and mobile ad hoc network.

**Bin Zan** is currently a Ph.D student in Electrical and Computer Engineering department at Rutgers University. He is working on location-aware systems, mobile networking, privacy and security under the guidance of Prof. Marco Gruteser at Winlab, Rutgers. He received his B.S. degree in Communication and Information Engineering department from University of Electronic Science and Technology of China (UESTC). He received his M.S. in Computer Science department from Clarkson University in 2006. Before joining Rutgers, he was working as software engineer in Adknowledge. Inc, Kansas City.

**Marco Gruteser** is an Associate Professor of Electrical and Computer Engineering at Rutgers University and a member of the Wireless Information Network Laboratory (WINLAB). He is a pioneer in the area of location privacy, having developed the first cloaking algorithms for spatial information. Beyond location privacy, his expertise includes location-aware networking and its connected vehicle applications. Previously, he was also a research associate at the IBM T.J. Watson Research Center, where he designed the software platform for the New York Times-featured BlueSpace smart office prototype. He completed a Vordiplom at Darmstadt University of Technology, Germany in 1998, received his MS and PhD degrees in Computer Science from the University of Colorado at Boulder in 2000 and 2004, respectively, and held a visiting position at Carnegie Mellon University. He has served on the technical program committees of numerous ACM and IEEE conferences, including MobiCom, MobiSys and INFOCOM. He also serves on the editorial boards of the journals IEEE Transactions on Mobile Computing and Elsevier Computer Networks. His recognitions include an NSF CAREER award, the I/UCRC Association's Schwarzkopf Prize for the ORBIT wireless testbed team, a MobiSys best paper award, and a Board of Trustees Research Fellowship for Scholarly Excellence at Rutgers University.

**Yanyong Zhang** has 10 years research experience in the field of distributed computing and performance evaluation. She is currently an Associate Professor in the Electrical and Computer Engineering Department at Rutgers University. She is also a member of Winlab. She has been involved in several NSF grants related to wireless networking, including the ORBIT wireless testbed, the PARIS project on privacy, and the NSF CAREER award that focuses on developing robust wireless sensor networks. She is an organizer of several ACM/IEEE workshops on self-managing systems. She has published over 45 papers on performance optimizations of various parallel and distributed systems, including sensor networks. She has also led WINLAB's efforts in performance evaluation of MANET protocols on ORBIT for the Army CERDEC. She has published over 50 papers in journals and conferences. She is a member of the ACM, the IEEE, and the IEEE Computer Society.