Performance of dead reckoning-based location service for mobile ad hoc networks

— Source link ↗

Vijay Kumar, Samir R. Das

Institutions: University of Cincinnati, Stony Brook University

Related papers:

- A scalable location service for geographic ad hoc routing

- Performance of dead reckoning-based location service for mobile ad hoc networks: Research Articles

- A distance routing effect algorithm for mobility (DREAM)

- GPSR: greedy perimeter stateless routing for wireless networks

- GPS-less low-cost outdoor localization for very small devices

Share this paper: f 𝕏 in ✉

View more about this paper here: https://typeset.io/papers/performance-of-dead-reckoning-based-location-service-for-41k452c4ah

Performance of Dead Reckoning-Based Location Service

for Mobile Ad Hoc Networks

Vijay Kumar

Department of Electrical & Computer Engineering

and Computer Science

University of Cincinnati

Cincinnati, OH 45221-0030

Email: `vkumar@ececs.uc.edu`

and

Samir R. Das

Computer Science Department

SUNY at Stony Brook

Stony Brook, NY 11794-4400

Email: `samir@cs.sunysb.edu`

Abstract

A predictive model-based mobility tracking method, called dead reckoning, is developed for mobile ad hoc networks. It disseminates both location and movement models of mobile nodes in the network so that every node is able to predict or track the movement of every other node with a very low overhead. The basic technique is optimized to use "distance effect," where distant nodes maintain less accurate tracking information to save overheads. The dead reckoning-based location service mechanism is evaluated against three known location dissemination service protocols: Simple, DREAM and GRSS. The evaluation is done with geographic routing as an application. It is observed that dead reckoning significantly outperforms the other protocols in terms of packet delivery fraction. It also maintains low control overhead. Its packet delivery performance is only marginally impacted by increasing speed or noise in the mobility model, that affects its predictive ability.

Keywords: mobile ad hoc networks, location-based protocols, rotuing protocols, performance evaluation.

1 Introduction

A mobile ad hoc network [25] is an autonomous system of mobile, wireless nodes. There is no static infrastructure such as base station. If two nodes are not within radio range, all message communication between them must pass through one or more intermediate nodes. Thus, in such a network, each node functions not only as a host but also as a router. As the nodes are mobile, the network topology is dynamic, leading to frequent and unpredictable connectivity changes. It is critical to route packets to destinations effectively without generating excessive overhead. This presents a challenging issue for routing protocol design since the protocol must adapt to frequently changing network topologies in a way that is transparent to the end user.

Knowledge of geographic locations of the nodes can aid in routing protocol design. Several location-based routing protocols have been proposed (see, for example, [21, 22, 20]) in recent literature. These protocols utilize available location information of other nodes for low-overhead routing. However, for these protocols to be effective, this location information must be efficiently disseminated and/or updated in the network. Location update can be purely opportunistic or passive (e.g., in LAR [21]), where location information is simply piggybacked on packets ordinarily exchanged by nodes. This makes the information stale very easily if the source-destination pair does not communicate very often, or if the destination node moves very fast. Either makes the "paging area" (geographic area where the destination can be located at the time of communication) very large, increasing control overhead. There are several protocols that use more active mechanisms. Broadly, they can be classified into two types — (i) location database systems that use some form of query-reply mechanism, such as SLURP [29] and GLS [23], (ii) location dissemination systems, that actively disseminate location information in the network, such as GRSS [16] and DREAM location service [2, 9].

Location database systems typically rely on one or more nodes in the network that work as location servers. The servers may be dynamically elected. They are updated proactively by moving nodes. These systems are inefficient when locations are frequently queried, as this increases the query-reply load. The query latency may also affect the performance of the application that is dependent on the location service. Also, elaborate schemes are needed to load balance the location service mechanism. In contrast, in location dissemination systems nodes proactively announce their locations to other nodes. While these schemes are simpler, they are usually inefficient as they are dependent on some form of periodic flooding.

In this work, we propose a distributed algorithm to build an efficient location dissemination system based on a technique called *dead reckoning*. Dead reckoning derives its efficiency from a predictive scheme that is used to track location of network nodes with little control overhead. Similar scheme has been used in cellular PCS systems before [24]. Our goal in this paper is to develop the dead reckoning-based loca-

tion dissemination scheme and evaluate its performance relative to some prominent location dissemination protocols.

The rest of the paper is organized as follows. In Section 2 we describe the dead reckoning technique in detail including some efficiency considerations. Section 3 includes description of the mobility models considered in our evaluation. Section 4 describes other prominent location dissemination protocols. Section 5 presents simulation-based performance evaluation of the dead reckoning technique. This section also presents comparative evaluation of dead reckoning-based location service with the other location dissemination protocols. Section 6 presents related work. We conclude in Section 7.

2 Dead Reckoning Technique

In most practical systems, a mobile node typically travels with a destination in mind. Even otherwise, its mobility pattern is unlikely to be a pure Brownian motion or random walk. We thus anticipate that in many situations, a mobile node's future location and movement will be correlated with its current location and movement – an observation also made in [24] in connection with PCS systems. Thus, location tracking schemes based on predictive models using current and past location and movement information are worth exploring.

In this paper, we take this approach using a method called *Dead Reckoning*. The original idea follows from an ancient navigation technique popularly used by sailors during the fifteenth century. This was before more accurate celestial navigation techniques were developed. In dead (or deduced) reckoning, the navigator found the ship's position by measuring the course and distance he had sailed from some known location, e.g., a port. The course was measured using a magnetic compass and the distance by estimating the speed and measuring the time. In a more general form of dead reckoning that we use in this work, a mobile node samples its own location continuously or periodically and constructs a model of its movement. The model can be simple or sophisticated, deterministic or stochastic, depending on the predictive ability of the mobile node. The node disseminates its current location as well as this model (together called the *dead reckoning model* or DRM) in the network. Every other node uses this information to track the location of this node. Very little location update cost is incurred if the model's prediction is accurate. In our knowledge, we are the first one to use dead reckoning for mobility tracking in ad hoc networks. We have used this technique earlier to develop new routing protocols [1]. It sould be noted that dead reckoning has long been considered to be a useful technique in other domains, such as distributed interactive simulations (see, for example, [7]).

We describe the details of the dead reckoning-based technique in the following subsections.

2.1 Location Estimation

For the purpose of this work, we use a very simple dead reckoning model which as a simple first order model that models a moving node's velocity (speed and direction). Complex models and use of path planning applications are possible, but not investigated here.

We assume that each node in the network is aware of its location. Most commonly the node will be able to learn its location using an on-board GPS (Global Positioning System) receiver. Other methods such as radio-location [6], beacons from an available fixed infrastructure [5], or GPS-less positioning systems such as [10] are also possible. The accuracy of the location information will affect performance, though we will not discuss it here. In the dead reckoning model we have used, each node constructs a movement model for itself by periodically sampling its location estimates. It then computes its velocity components v_x and v_y along the X and Y axes from two successive location samples (x_1, y_1) and (x_2, y_2) taken at times t_1 and t_2. Thus,

$$v_x = \frac{x_2 - x_1}{t_2 - t_1}$$

and

$$v_y = \frac{y_2 - y_1}{t_2 - t_1}.$$

After the first calculation of its velocity components, the node floods the network with this information using a *DRM update* packet. This packet contains the node's id, its location coordinates and the calculated velocity components.

The location and movement models together form the dead-reckoning model. Each node maintains a DRM table. Whenever it receives a DRM update from another node it adds or updates an entry for that node's model in its *DRM table.* The model entry for each node in the table has a timestamp denoting when it was last updated. Thus, each node now has a location and movement model for every other node in the network. It uses this to estimate the location (x_{est}, y_{est}) of the nodes at the current time as per the following formula:

$$
\begin{aligned}
x_{est} &= x_{mod} + (v_{x_{mod}} * (t_{cur} - t_{mod})) \\
y_{est} &= y_{mod} + (v_{y_{mod}} * (t_{cur} - t_{mod})),
\end{aligned}
\tag{1}
$$

where, (x_{mod}, y_{mod}) are the (x, y) coordinates and $(v_{x_{mod}}, v_{y_{mod}})$ are the velocity components of the node in the model table, t_{mod} = time at which model was updated and t_{cur} = current time.

Note that we have a tacit assumption here that t_{mod} actually reflects the time when the model is computed at the originating node. It is possible to include this computation timestamp in the DRM update, and use this time as t_{mod} instead of the update time. In that case, however, t_{mod} and t_{cur} would be timestamps

taken on different nodes. So, for this scheme to work, the nodes must have access to synchronized clocks on all nodes. Synchronized clocks are not unrealistic as they can be obtained from GPS fixes. But even if such synchronization is not available, we expect that error introduced would be negligible unless the mobile nodes move at an unrealistic fast speed. The reasoning for this is as follows.

(i) Assuming that t_{mod} is the computation time, "loosely" synchronized clocks will work fine so long as the time-scale of the clock error is much smaller than the time-scale of a node's movement to significant distances. A significant distance here is a distance larger than the radio range.

(ii) If no clock synchronization is available at all, t_{mod} should reflect the time when the DRM update is recorded. In this case, the speed at which update messages propagate in the network must be much faster than the speed of the mobile nodes.

After the initial distribution of its dead-reckoning model, each node continues to periodically sample its location (x_{cur}, y_{cur}) and also computes its predicted location as per the above model it advertised. It calculates the deviation of its current location from its predicted location by simply computing the Euclidean distance

$$d = \sqrt{(x_{cur} - x_{est})^2 + (y_{cur} - y_{est})^2}.$$

If this distance d exceeds a predetermined threshold (called the *dead reckoning threshold*) the node recalculates the DRM and disseminates it again in the network. Otherwise, no further dissemination takes place.

The threshold essentially determines the allowable error in the location estimates. Too small a threshold results in too many updates adding to the load in the network. Too large a threshold results in inaccurate and stale location information persisting the network for a long time. It is expected that the application (e.g., routing) that utilizes the location information is able to choose an appropriate threshold based on the desired level of accuracy.

2.2   Efficient Dissemination

The basic technique can be upgraded for efficiency in large networks. The idea is to disseminate more accurate location information to nearby nodes and progressively less accurate information to far away nodes. This reduces control overhead. Most applications (such as routing) will tolerate less accurate location information for far away nodes (sometimes called "distance effect" [2]). A layering technique is proposed to achieve this.

A mobile node N views the entire geographic area as divided into concentric circular layers with itself at the center. Let us denote the layers as L_1, L_2, L_3, with the subscripts progressively increasing outward (see Figure 1). For simplicity, the layers are assumed to be of the same width W. (We use this simplifying
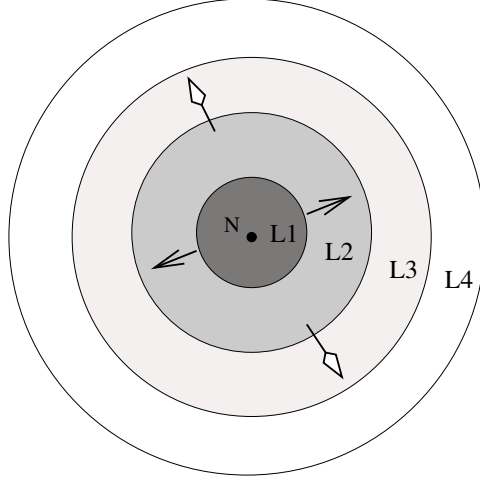
Figure 1: Concentric layers with the mobile node N at the center. Layering is not absolute; it is relative to a specific node (N). Each node has its own layer structure. The nodes in the network can reside in different layers with respect to different other nodes.

assumption in this paper; however, in general, the layer widths can be different.) Different thresholds are maintained for nodes in different layers, with the thresholds progressively increasing in the outer direction. The thresholds of different layers are chosen such that the "angular error" threshold (maximum allowable angular location error) remains constant for all layers. From simple trigonometry, this requires that the threshold used in layer L_i is $i \times$ threshold for the innermost layer (L_1), assuming that the layer widths are the same. The threshold for the innermost layer is the only input parameter to the algorithm.

To implement the layering technique, each mobile node must maintain state information regarding the DRM updates propagated to nodes in different layers. Note that not all layers get all updates. It must also tag each update with a layer number before which an update flood must be stopped. This sounds straightforward, but this technique presents a problem. If the mobile node determines that nodes in a layer L_i require an update, there is no way to send the update only to nodes in layer L_i. All nodes in layers $L_1, \ldots, L_{i-1}$ must also receive and propagate the update. This is an unnecessary overhead.

Figure 2 demonstrates this problem. Assume that the mobile node is moving in a straight line in the direction shown. Assume also that a network wide update has been sent at time $t = 0$, when the node is at A and the model used in this update predicts that the node is stationary. Assume that the DRM threshold for the innermost layer (i.e., L_1) is δ. When the node moves past B at time $t = \tau_1$ an update must be sent to all nodes in layer L_1. Assume that this update also models the mobile node as stationary. When the node moves past C at $t = \tau_2$ another update must be sent to nodes in layers L_1 and L_2. Assume that enough samples have been gathered by this time for the movement model to be correctly predicted. Thus, when the node reaches D at $t = \tau_3$, the update needs to be sent to nodes in layer L_3 only. Nodes in L_1 and L_2
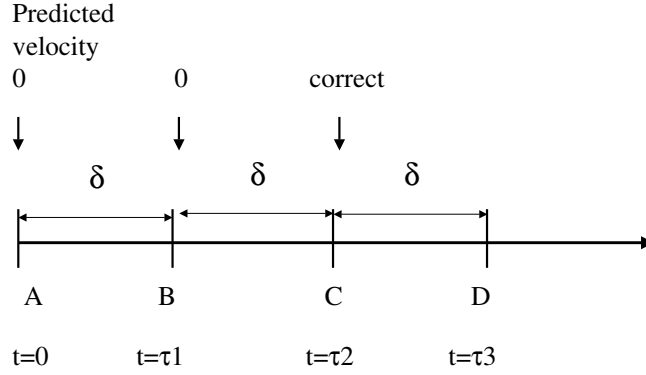
Figure 2: Example of buffering technique.

already have the correct model of the node's movement. Now, how do we send updates to layer L_3 nodes directly skipping over nodes in L_1 and L_2? So nodes in L_1 and L_2 will get updates as well! These updates are redundant, as those nodes already have the correct model of the mobile node's movement.

To get around this additional overhead, we maintain the state information in the "boundary" nodes between two layers instead of at the mobile node in question. The idea here is to "buffer" a DRM update in a boundary node that "stops" that update,[1] and then propagate that update from the boundary nodes towards outer layers, if this update is needed in future by the outer layers. As a rule, a boundary node (say, P) always remembers the DRM update from a mobile node (say, N) that P last propagated. We refer to this by the term "old DRM," while the latest DRM update from N that P stops and buffers (i.e., does not propagate) is called "current DRM" or "buffered update." The old DRM serves as a reference to the DRM propagated to the outer layer nodes. If, at any time, the deviation of the mobile node's location as per the old and current DRMs is more than the next outer layer's threshold, the buffered DRM update (same as the current DRM) is propagated to the next outer layer. Note that only one current DRM and one old DRM need to be maintained on each node for every other node in the network.

In the above example, at $t = \tau_3$ the nodes at the boundary of L_2 and L_3 realize that the current model that they previously buffered must now be propagated to L_3. This is because at $t = \tau_3$ the old model predicts the mobile node to be at A whereas the current model predicts it to be at D, a deviation of $3 * \delta$.

---

[1]Note that a boundary node is not always exactly at the boundary. A boundary node between layers L_i and L_{i+1} is in fact in L_{i+1}, but is able to receive transmissions from one or more nodes in L_i. This "loose" definition of boundary nodes will work fine for our purpose, and is efficient to implement. If an update is meant for layer L_i, a boundary node between L_i and L_{i+1} will receive this update, but will not propagate any further. We refer to this as "stopping" the update.

2.3 Implementation of Efficient Dissemination

Note that the layers are node specific. This means that each node N has its own layering. Layers move as the node N moves. A node P can be at L_i with respect to node N (depending on its distance from node N) and at a different layer L_j with respect to another node M.

To implement the "layered" technique described above, each node in the network maintains two DRM tables — one "old" and the other "current." Each table is linear in the size of the network, having at most one constant-sized DRM model entry per node in the network. Note that a node P with a valid entry in the old DRM table for another node N is a boundary node for some layer of N (depending on the percieved distance between P and N), and vice versa.

The technique involves these major components:

1. A mobile node (say, N) generates DRM updates *only* for the innermost layer (layer L_1). The boundary nodes between L_1 and L_2 always stop and buffer this update.

2. If any node P stops and buffers an update from N (same as "current DRM") by virtue of being a boundary node between two layers for N, it is P's responsibility (i) to remember the old DRM it last propagated (it simply copies the current DRM model entry of N to the old DRM model entry and then updates the current DRM model entry by the received update), and (ii) to monitor when N's location deviation as per the old and current models exceeds the DRM threshold of the next outer later. When that happens, P must propagate the buffered update (current DRM) outwards to the next higher layer.

3. When a node P decides to propagate a buffered update from N to the next higher layer for N, the update is again stopped and buffered at the next boundary. This process continues for all layers. Updates can remain buffered for negligible or even zero time; they may be buffered indefinitely, or can be overwritten by other newer updates.

One detail that remains to be discussed is the effect of node movement across layers. Nodes moving to outer layers with respect to another node (say, N) do not present a problem as they have more accurate location information for N than desired. However, the scenarios where nodes move to inner layers demand attention, as then they have less accurate location information than desired. Such nodes can simply update the location information for N from a neighbor. If a boundary node moves to an inner layer, we postulate that it must not propagate its buffered update any more, as this update can potentially "pollute" location information in other nodes with less accurate information. Such nodes simply discard the buffered update. If all boundary nodes for a layer move out to other layers, it does present a problem, however — as all copies of the buffered updates are lost. But we anticipate that this situation will be rare. Even if it does happen, it is no worse than lost messages in the wireless links. In this context, note that message losses in

the wireless medium do reduce the efficacy of the protocol, as then "stale" location information is used. In our simulations we have used realistic wireless channel and link layer protocol models such that losses due to channel errors and multiple access interference are taken into account.

3   Mobility Models

Since we are using a predictive model for location tracking, success clearly depends on the accuracy of prediction. Routing load will increase greatly with inaccurate predictions making the whole scheme inefficient. It is hard to make general comments on accuracy, as realistic usage models for mobile ad hoc networks are still unclear. Also, models generated directly from applications (such as path planning) can greatly improve the performance of the dead reckoning technique relative to the case when mobility is predicted merely by taking location samples.

    We will evaluate our technique via extensive simulations in section 5. Part of our evaluation will use noise in the mobility model to affect the predictive property of dead reckoning. For completeness, here we review the mobility models we will use in our evaluation.

3.1   Random Waypoint

The random waypoint mobility model [4] is an extension of a traditional random walk model that is used to model many physical systems. In random waypoint, a mobile node moves towards a destination location with a randomly chosen speed; pauses for a specific period of time (pause time) when it reaches this destination; and then chooses a random destination again. See Figure 3.1(a). All destinations are randomly chosen from within a predefined area. A node maintains constant speed while moving, which helps building the dead reckoning model in the micro-level. But depending on the speed, pause time, and size of the field, the model of a node may change quite frequently. The model is memory-less; the speed and direction after a pause are completely independent of the speed and direction before the pause. Almost certainly DRM updates will be necessary both after a pause and again after restart unless the pause is fairly short relative to the location sampling interval.[2] For very brief pause, only one update will be necessary per pause as then the drift from the predicted trajectory after the pause will not be significant enough to trigger an update.

3.2   Gaussian Markovian

Let us contrast the above model with another mobility model popularly used in cellular PCS networks that presents more realistic random, but correlated movements where correlation can be tuned with a single

---

[2]In most practical implementations the location of a node will be sampled at small, periodic intervals, instead of continuously.

(a) Random Waypoint   (b) Gaussian Markovian Random Waypoint
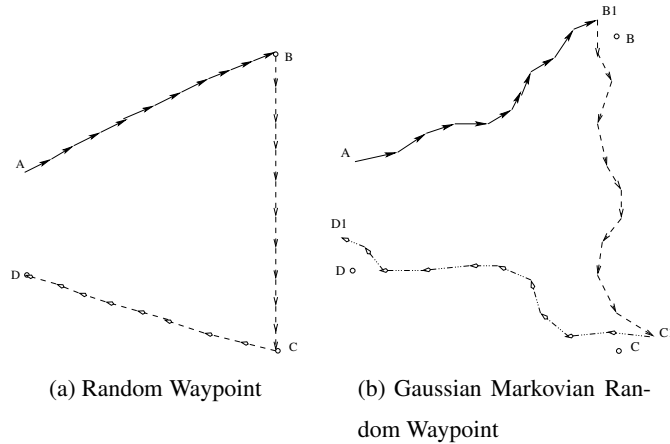
Figure 3: At A, the mobile node selects random destination B, initial speed and direction. In random waypoint, it travels in a straight line with constant speed towards B. In the Gaussian-Markovian extension, the travel is noisy, changing speed and direction at small intervals, but still travelling in the same general direction, with the same average speed. Note that the node may not reach the intended location (say, B), but reaches a neighborhood location (say, B1).

parameter. This is the Gauss-Markov mobility model [24, 8]. Initially each node is assigned a current speed and direction. At fixed intervals, the speed and direction are updated. Specifically, the speed and direction at the nth instance is calculated based upon the value of speed and direction at the $(n-1)^{th}$ instance and a random variable using the following equations:

$$s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{1-\alpha^2} s_{x_{n-1}}, \tag{2}$$

$$d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{1-\alpha^2} d_{x_{n-1}}, \tag{3}$$

where s_n and d_n are new speed and direction of mobile node at time interval n; $0 \leq \alpha \leq 1$, is the tuning parameter used to vary the randomness; $\bar{s}$ and $\bar{d}$ are constants representing the mean value of speed and direction as $n \to \infty$; and $s_{x_{n-1}}$ and $d_{x_{n-1}}$ are random variables from a Gaussian distribution. By setting $\alpha = 0$ we get a very random motion, while $\alpha = 1$ gives a completely linear motion. Intermediate levels of randomness are obtained by varying the value of α between 0 and 1.

3.3 Gaussian Markovian Random Waypoint

We will use a hybrid of the above two models for studying the effects of mobility on dead reckoning. The idea is to use random waypoint at the macro level and extend it using the Gaussian-Markovian technique at the micro level to provide a noisy behavior. The hope is to be able to model realistic mobility behaviors. For

10

example, when we a drive to a location, we do not follow a straightline path, rather go towards the general direction with twists and turns. Even a planned path may need to be modified based on traffic conditions. The speed, similarly, may not be exactly constant, but varies around a chosen average speed. In addition, we may not always reach a specifically chosen coordinate, but a general neighborhood around that coordinate.

Thus, in this model, the mobile node still chooses a random destination location and speed, and then computes the direction of travel. However, it follows the model presented in equations (2) and (3) to travel in this direction in small time steps. The model is initialized with s_0 and d_0 being equal to the chosen speed and direction. $\bar{s}$ and $\bar{d}$ are the mean speeds and direction computed on line by averaging over all timesteps. The node travels following this model for an amount of time (say, T) equal to what it would take for it to reach the current destination using the chosen speed. Note that it may not now reach the destination, but possibly a location close to it depending on the amount of noise. After the time T the node pauses according to a given pause time, chooses a random destination location, and repeats the above process. See Figure 3.1 for illustration of the random waypoint model and its Gaussian-Markovian version.

The noise can be controlled by varying α. $\alpha = 1$ means a pure random waypoint model; $\alpha = 0$ signifies a model with significant noise (purely Gaussian behavior without any memory for each time step). The predictive ability of dead reckoning will reduce with increasing α. A set of performance results will be presented in section 5 by varying α. In our evaluation, the means of the Gaussian variables s_x and d_x are always zero and standard deviations are 1 m/sec and 0.4 radian (23 degrees) respectively. Note that this is a significant angular deviation.

4   Other Location Dissemination Protocols

We will evaluate dead reckoning against some prominent existing techniques in section 5. For the readers' benefit and for the sake of completeness, we briefly describe them here.

4.1   Simple Location Service

A node using the Simple Location Service protocol[9] transmits a location packet to its neighbors at a given rate, which adapts according to location change. A location packet is transmitted every $\min\{\frac{R}{\alpha v}, X\}$ seconds, where R is the transmission range, v is the average velocity of the mobile node, α is a scaling factor, and X is a predetermined constant. The best values of various constant parameters have been found via numerous simulations in [9].

Each packet in this service contains upto E entries from the node's location table; the entries are chosen from the table in a round robin fashion. Thus, with each transmission of a location packet, upto E entries are exchanged between neighbors. As multiple packets are transmitted, all location information in the node's

knowledge will be shared with its neighbors. A node always maintains the latest location information of other nodes.

4.2 The Distance Routing Effect Algorithm for Mobility (DREAM)

The DREAM protocol [2] uses location information for routing. Here, we focus on the location service aspect of DREAM and not the routing aspect. The DREAM location service is based on the following two basic principles:

1. The faster a node moves, the more often it must communicate its location.

2. The greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Thus, nodes that are far apart, need to update each others locations less frequently than nodes that are closer.

The exact details of implementing the location service are lacking in the original DREAM paper [2]. However, authors in [9] developed the details of the algorithm following the DREAM philosophy. Each node periodically transmits a location packet that contains the node's location, speed and a timestamp. Each mobile node in the ad hoc network broadcasts location packets in the neighborhood at a given rate, and to other distant nodes at another, lower, rate. In [9], neighborhood has been chosen as one hop, and good performance has been reported with this choice. So we will use this idea in our evaluation. Finally, the rate at which location packets are transmitted to neighborhood nodes can be derived in a similar fashion as in Simple above [9]. Location packets to distant nodes can be transmitted every Y packets transmitted to neighborhood nodes or every Z seconds. Good values of the constant parameters have been found using many simulations in [9].

4.3 Geographic Region Summary Service (GRSS)

In GRSS [16], the geographic area of the network is divided into a grid with small squares (called order-0 squares). Four adjacent order i squares form an order $i + 1$ square, thus generating a hierarchy of squares of different orders and sizes. A mobile node is located within exactly one square of each order. Nodes residing in the same order-0 square use a local routing protocol to exchange their locations and neighboring node lists with each other. The ultimate goal is for every node to know the exact paths to all the other nodes in the same order-0 square.

Nodes close enough to the square boundaries receive routing messages from neighboring nodes in the adjacent squares. Such nodes are called "boundary nodes." Every boundary node in an order-0 square generates a *summary* about the nodes in the square and sends it as a summary update message to other
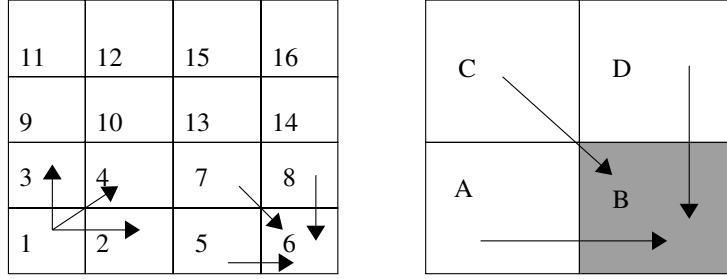
Figure 4: The entire network is divided into squares. Squares of two orders are shown. Summary information is exchanged between sibling squares, shown by arrows. Only some arrows are shown to reduce clutter.

"sibling" order-0 squares. Sibling order-i squares are the squares that make up an order-$i + 1$ square. The summary message is actually a list of all the nodes present in that particular square alongwith the location co-ordinates of the center of that square. Similarly, summary messages are generated by boundary nodes in order-1 square and so on, continuing recursively until the highest order square is reached (see Figure 4). Since every node resides in squares of all different orders, the union of all received summaries covers all nodes in the network except nodes in its own order-0 square. For the former nodes, the current node has information about some square(s) the node is located in (the center of the least order square for which information is available is approximately used as the node's location); for the latter, the current node has exact location information via the local routing protocol. Notice that by the nature of the protocol better approximations (lower order squares) are available for nodes that are in the vicinity. Accurate information is available for nodes that are in the same order-0 square.

The author of the protocol has used GRSS to perform geographic forwarding [16]. The available location information is used to make forwarding decisions at each intermediate node, and the packet is forwarded to the neighbor closest to the destination. Note again that better approximations are available as packets get closer to the destination. When the packet reaches the same order-0 square as the destination, the exact path is available via the local routing protocol.

The advantage of GRSS is that individual node locations are not flooded in the entire network. Summary information is exchanged instead, that too only between sibling squares of every order. Beaconing to implement the local routing protocol and summary updates are done at periodic intervals.

It is important to note here that similar to Simple and DREAM, the GRSS protocol also carries parameters that need to be set appropriately for efficient operation. While the original GRSS paper [16] does not mention them explicitly, we identified the following parameters: (i) *Neighbor Time Out (NTO)*: The timeout period to remove a neighbor entry in the local routing protocol if no beacon is received, (ii) *Grid Size (GS)*: The size of the lower most square, (iii) *Summary Period (SP)*: Time interval to send out summaries. More

words about parameter setting will come later.

5   Performance Evaluation

5.1   Simulation Platform

We have used *GloMoSim* [14], a discrete event simulation tool with wireless networking modeling library developed at UCLA. We have evaluated ns-2 [11] simulator as well, as a significant number of ad hoc networking researchers use ns-2. However, we favored GloMoSim for its better scalability as we plan to simulate large networks for our study (not reported here). The MAC layer model uses the DSSS PHY reference configuration of the IEEE 802.11 standard [17]. The radio model reflects the behavior of the first generation WaveLan radio interface cards (2 Mbps). For the chosen radio parameters (somewhat different from the default GloMoSim parameters) the nominal radio range is about 250 meters.

For performance measurements, we use geographic routing as an application that requires the use of location dissemination. In essence, performance of routing is indicative of the efficacy of the location dissemination. In geographic routing, we use greedy forwarding, where any node P forwards the data packet to the neighbor which is nearest to the location of the destination of the packet. Location of the destination is according to the knowledge of P. Greedy forwarding, however, has a drawback. It cannot route around "holes" or "voids" [20]. But we will use dense networks in our simulations where such possibility is poor.

For traffic model, 30 data sources are used to send 512 byte packets. Traffic sources are constant bit rate (CBR). The sessions are long-lived; they start towards the beginning of the simulation and stay until the end. The source-destination pairs are spread randomly over the network. Mobile nodes move with an average speed chosen randomly between 0-10 m/s other than the experiments where we vary the speed. Experiments are always run for 900 seconds for various number of nodes between 100 and 200. The area in which the mobiles move about change with the number of nodes with the spatial node density approximately constant. The pause time is varied from 0 seconds to 900 seconds. 0 sec pause time means continuous mobility and 900 seconds pause means stationary network. Each data point in the plots represents an average of at least 10 runs with identical traffic models, but different randomly generated mobility scenarios.

The location sampling interval for dead reckoning model is kept constant at 1 second. A set of experiments (not reported here for lack of space) was conducted to choose an appropriate layer width. It was noted that larger layer width (and hence smaller number of layers) always increased routing load, provided the distant nodes with more accurate location information, but did not appreciably improve geographic routing performance. Thus, small layer widths are preferable. Thus, we set layer width to be same as the radio range (250m). Narrower layers would be somewhat meaningless.

The following performance metrics are evaluated: (i) *Packet delivery fraction*: ratio of the packets
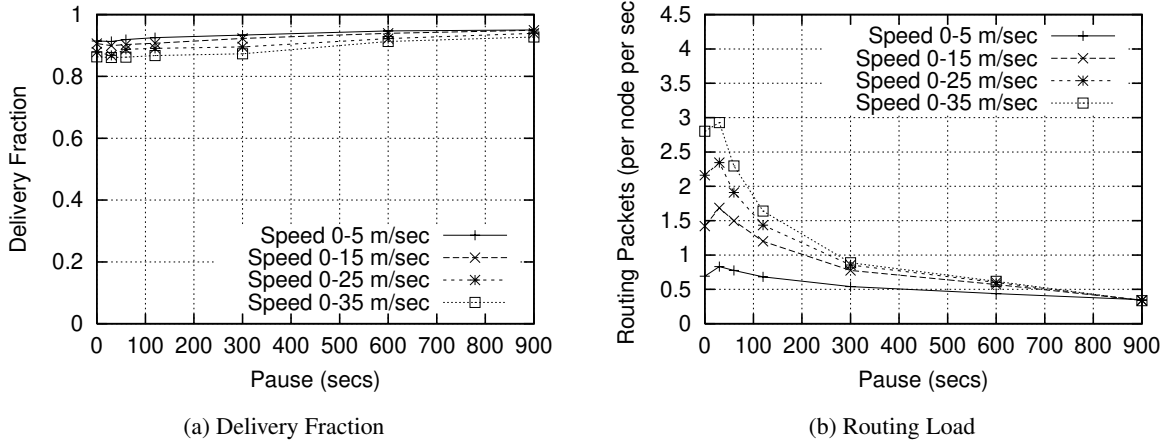
Figure 5: DRM performance with varying node speed in a 100 node network for the random waypoint mobility model.

delivered to the destination to those generated by the CBR sources. (ii) *Routing load*: the number of control packets transmitted per node per sec to maintain and propagate location information. For experiments comparing different protocols, routing load is also measured as the number of bytes per node per sec as control packet size differs across protocols.

5.2 Performance of DRM-based Location Service

In this subsection we evaluate the performance of DRM-based location service, and study how various mobility parameters such as speed and noise may affect performance. The performance results were obtained for a network of 100 nodes in an area of 1000m $\times$ 1000m. The packet rate is 2 packets per second. The DRM threshold is kept at 15 m. Smaller threshold improves performance, but also increases simulation run times, on the other hand, produces qualitatively similar results for the effect of noise and speed.

Figure 5 shows performance with varying speed of the mobiles. The speed has been varied from 0-5m/s to 0-35m/s, and the random waypoint model is used. With increase in speed The delivery fraction degrades only marginally, though routing load increases substantially, particularly for small pause times (higher mobility). This is expected as higher speed means more frequent pauses and subsequent changes in direction. However, note that increase in routing load is not proportionate to the speed. This is because the deviations after a pause/direction change may not always be subsantial to trigger a location update. Also, for larger pause times, the pause times dominate the simulation time (in contrast to movement time).

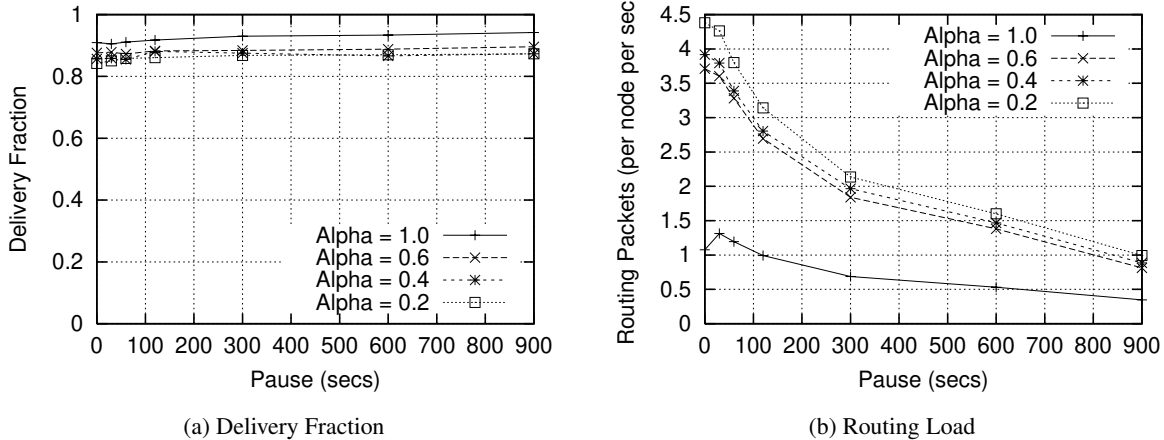Next we use the Gaussian Markovian Random Waypoint model to evaluate the efficacy of the predictive

(a) Delivery Fraction   (b) Routing Load

Figure 6: Effect of noise in the mobility model for a 100 node network. The Gaussian Markovian random waypoint mobility model is used with α as the noise paramater.

model in presence of noisy motion. Recall that the parameter α controls the noise, $\alpha = 1$ is equivalent to pure random waypoint with straightline motion. Figure 6 shows the performance with different α values. The speed of the nodes here is fixed at 0-10m/s. As expected, the routing load increases with decreasing α (increasing noise), as more noise demands more frequent DRM updates. Data forwarding is mostly able to keep up, however. The delivery fraction is only marginally affected with increasing noise.

5.3 Performance Comparison with Other Location Dissemination Protocols

In this subsection our goal is to compare the performance of the DRM-based location service with other existing location dissemination protocols. We have chosen three prominent protocols to compare with, *viz.*, Simple, DREAM and GRSS. They have been reviewed in section 4. One problem we faced in developing simulation models of these three protocols is that they all have parameters that need to be tuned to extract the best possible performance. This is necessary for a fair comparison. The parameters for Simple and DREAM were investigated in [9]. While their work was helpful, our network being much larger, we did independent set of tuning experiments to come up with a set of parameters that provide the best performance. The best set of parameters was found to be somewhat different from those obtained in [9]. Similar effort was expended for GRSS as well. Our own protocol DRM also needed tuning to select an appropriate value of the DRM threshold. This was a relatively simpler process, however, as there was one single parameter to tune.
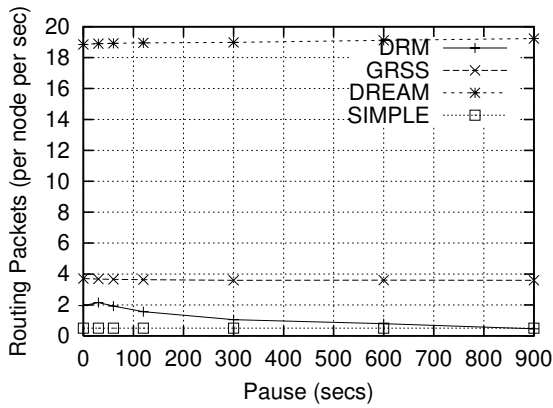
Experiments are conducted for 100 and 200 nodes networks in an area of 1000m × 1000m and 1500m × 1500m respectively. For each of the above set up 30 CBR data sources are used; 512 byte data packets
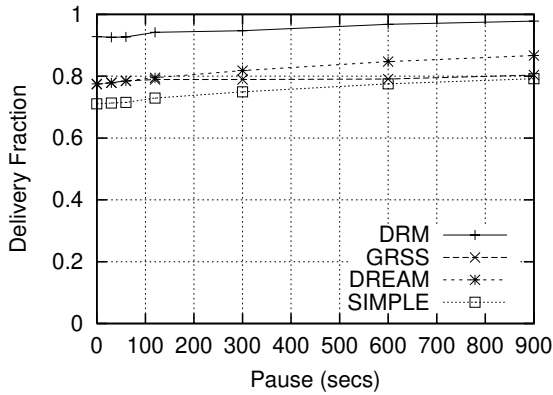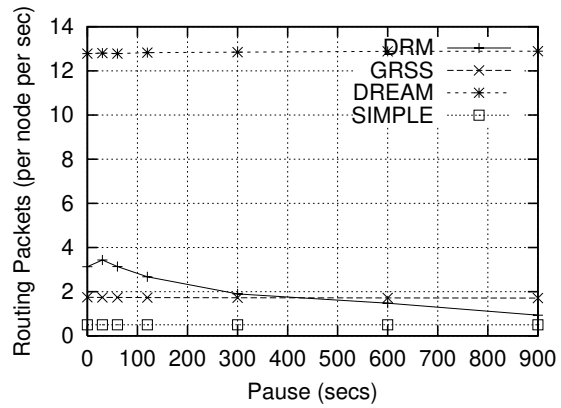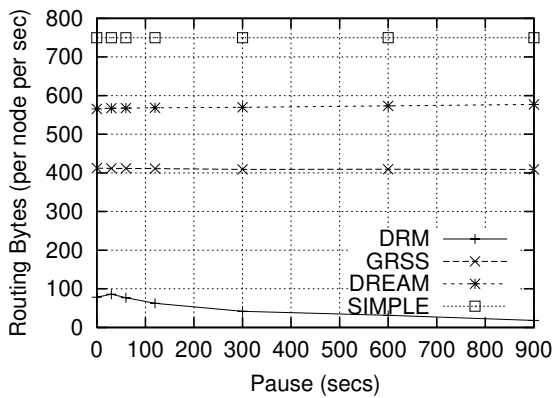
(a) Delivery fraction, 100 nodes
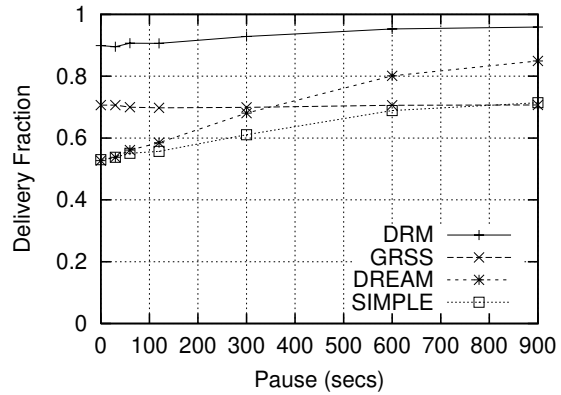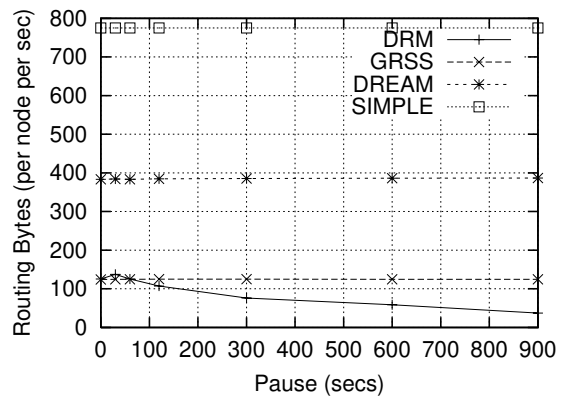
(b) Delivery fraction, 200 nodes

(c) Routing load in packets, 100 nodes

(d) Routing load in packets, 200 nodes

(e) Routing load in bytes, 100 nodes

(f) Routing load in bytes, 200 nodes

Figure 7: Delivery fraction and routing load performance of DRM relative to other location dissemination protocols.

are sent at a rate of 1 packet per second. Note that the packet rate is chosen lower than before so as not to overwhelm the larger 200 node network. The mobility model uses random waypoint ($\alpha = 1$) with average node speed chosen between 0-10m/s. It was felt that experiments with Gaussian-Markovian random waypoint model ($\alpha < 1$) was not necessary. The performance of the DRM-based technique with higher noise in the model would certainly degrade; however, as the performance results will demonstrate, the degraded performance of DRM (as per the results from the previous subsection) is still better than the performance of the competing protocols.

For a reader interested in reproducing the results of the simulation, the protocol parameters used are as follows. For the 100 (200) node network, in the Simple protocol, $E = 70$ (110), $\alpha = 2$ (2) sec, $X = 24$ (24); in DREAM, $\alpha = 2$ (12), $X = 5$ (15), $Y = 13$ (13); in GRSS, GS = 450 (450) m, NTO = 1 (2) sec, SP = 2 (4) sec; in DRM, DRM threshold = 3 (3) m.

The plots are as shown in Figure 7. Two sets of routing load plots are shown, in terms of packets and also in terms of bytes, as routing packet sizes differ widely across protocols. The DRM-based protocol provides the best delivery fraction for both 100 and 200 node networks, significantly ahead of the other protocols. It also has low routing load in terms of both packets and bytes. Though the Simple protocol has a low packet overhead, it has much more byte overhead as it exchanges partial location tables. DREAM does very poorly in both packet and byte overheads. Large routing load is the reason these two protocols perform poorly in packet delivery performance. Large routing load leads to congestion which translates to collisions and/or channel access delays for data packets. This leads to data packet losses from the network interface buffer. Congestion also leads to loss of routing packets for similar reasons, leading to stale location information that in turn affects data packet delivery performance.

GRSS routing performance is comparable to our DRM-based protocol in the 200 node case, though it still has a higher overhead in the 100 node case, albeit not as high as DREAM and Simple. However, its packet delivery performance is still poor. We found that routing loops are formed in GRSS due to the failure of the greedy geographic forwarding, though this has been rarely the case for other protocols in the scenarios we have used. We conjecture that this is due to the way grid partitioning is done in GRSS, where two nearby nodes fall into two different order-0 squares. So they have only approximate location information about each other. This makes failure of greedy forwarding likely. This also makes the routing path longer. Longer paths raises the possibility of packet losses as it raises the network load. Note that our experience is different from the performance reported in [16], where the author of the protocol used very short simulation run and very low data load by way of using short-lived data sources.

For lack of space we do not report experiments here that vary speed and noise in the mobility model. However, the reader will note that the delivery fraction for the DRM-based protocol only marginally degrades (upto about 5%) with increase in speed and noise (Figures 5 and 6). In Figure 7 DRM's delivery

18

performance is at least 10% higher relative to the next best protocol. So we believe that DRM will still significantly outperform the other protocols, even when speed and noise are increased.

6 Related Work

There is a growing body of work that uses location information for routing in ad hoc networks. Early work started with the LAR and DREAM protocols. LAR [21] works as an add-on to any on-demand flood-based protocol such as DSR [19] or AODV [26], and limits the route request flood within a zone where the destination is most likely to be found based on its past location and speed information. The location dissemination here is passive. DREAM [2], on the other hand, disseminates location information actively (periodically) and creates a location database locally. It then floods data packets within a cone radiating from the source, where the destination is guaranteed to be found within the broad end of the cone. The design of this cone follows from the available location information. (Note here that the DREAM routing protocol was not used in our study above; only the DREAM location service component was used, as our goal was to compare location service protocols with routing as an application.)

Several techniques were proposed for geographic forwarding where availability of location information is assumed and this information is directed used for packet forwarding without having to explicitly establish a route. A very simple forwarding scheme simply forwards the packet to the neighbor closest to the destination [12]. This is the technique we used. As mentioned before, some techniques are needed if this greedy method reaches a dead end, i.e., there is no neighbor closer to destination than the current node itself. Karp and Kung in their GPSR technique [20], and Bose et. al. [3] independently, present efficient techniques to recover from such situations. Another solution is provided in [18].

In [23] authors propose GLS, a location database service. GLS uses geographic hierarchy to partition the network space onto grids. Each node then maintains its current location in a number of location servers distributed throughout the network. Queries and registrations are sent to these servers using geographic forwarding. Care is taken such that location queries are sent to servers closer to the querier. Other forms of similar location database services are Quorum Based Service [15] and Homezone [13] and [27]. In Quorum services, location updates are sent to a subset (quorum) of available nodes, and location requests are referred to a potentially different subset. When these subsets are designed in such a way that their intersection is non-empty, it is ensured that an up-to-date version can always be found. In Homezone services, the concept of a virtual homezone is used, where a node's location information is stored. In [29], Woo and Singh present SLURP (Scalable Location Update-based Routing Protocol) based on the concept of Mobile IP. Nodes are assigned home regions and all nodes within a home region know the approximate location of the registered nodes. As nodes travel, they send location update messages to their home regions.

Prominent location dissemination services have already been described in section 4 and are not described here.

Finally, Liang and Haas's predictive model-based mobility management technique [24] is very similar to our base dead reckoning technique, but has been targeted for cellular PCS networks. Here, location updates are sent to wireline backbone network when the mobile's predicted location differs from its actual location by more than a threshold distance. The predicted location is used by the wireline backbone to compute a paging area for the mobile, when a call arrives. The authors here use the Gauss-Markov model as described earlier, and analytically demonstrate that, for correlated movements, the predictive model incurs much lower mobility management costs compared to a purely distance-based scheme.

In somewhat similar vain to our routing technique Su [28] has proposed a mobility prediction method that uses the location and mobility information provided by GPS to predict the link expiration time between nodes and perform rerouting when a route breaks. They have applied this method to various classes of ad hoc routing protocols, viz., on-demand unicast routing, distance vector routing and multicast routing. Finally, in [8] the authors reviewed several mobility models for ad hoc networks including the Gaussian-Markovian technique we used. Location information services including Simple and DREAM have been evaluated in [9].

7   Conclusions

We have developed a location dissemination protocol called dead reckoning that uses predictive models to track node locations. Significant deviation from the predicted model triggers location updates. Better prediction keeps control overhead lower. The basic protocol has been optimized to maintain a more accurate location information on nodes closer to a mobile node relative to farther nodes by using a layering and buffering technique. This improves control overhead even further.

We have evaluated the performance of this protocol against three other significant location dissemination protocols. We have chosen geographic routing as an application for performance evaluation. Dead reckoning demonstrates a much better packet delivery performance relative to the other protocols with a moderately low routing load. In addition, its packet delivery performance is not significantly affected by speed or noise (randomness) in the mobility model. As a by-product of this study we developed a new hybrid mobility model by extending the popular random waypoint mobility model by Gaussian-Markovian behavior that let us control the noisy behavior in the mobility model.

Our on-going work consists of studying the behavior of dead reckoning for very large networks (about 1000 nodes), developing analytical methods to determine appropriate threshold value, and evaluating the protocol performance relative to location database protocols.

References

[1] A. Agarwal and S. R. Das. Dead reckoning in mobile ad hoc networks. In *Proc. of 2003 IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, March 2003. To appear.

[2] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. Distance routing effect algorithm for mobility (DREAM). In *4th International Conference on Mobile Computing and Networking (ACM MOBICOM'98)*, pages 66–84, October 1998.

[3] P. Bose, P. Morin, P. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the DIAL-M'99 Workshop*, Aug. 1999.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th International Conference on Mobile Computing and Networking (ACM MOBICOM'98)*, pages 85–97, October 1998.

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[6] J. Caffery and G. Stuber. Overview of radiolocation in CDMA systems. *IEEE Communications Magazine*, 36:38–45, April 1998.

[7] W. Cai, F. B. S. Lee, and L. Chen. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In *Proccedings of the 13th Workshop on Parallel and Distributed Simulation (PADS)*, pages 82–89, 1999.

[8] T. Camp, J. Boleng, and V. Davies. Mobility models for ad hoc network simulations. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[9] T. Camp, J. Boleng, and L. Wilcox. Location information services in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3318–3324, 2002.

[10] S. Capkun, M. Hamdi, and J.P. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proc. Hawaii Int. Conf. on System Sciences*, Jan. 2001.

[11] Kevin Fall and Kannan Varadhan (Eds.). *ns* notes and documentation, 1999. available from http://www-mash.cs.berkeley.edu/ns/.

[12] Gregory Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, ISI, 1987.

[13] S. Giordano and M. Hamidi. Mobility management: The virtual home region. Technical report, October 1999.

[14] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[15] Z.J. Haas and B. Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, April 1999.

[16] Pai-Hsiang Hsiao. Geographical region summary service for geographical routing. *Mobile Computing and Communications Review*, 5(4), 2001.

[17] I.E.E.E. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE standard 802.11–1997, 1997.

[18] Rahul Jain, Anuj Puri, and Raja Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications Magazine*, Feb 2001.

[19] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile computing*, chapter 5. Kluwer Academic, 1996.

[20] Brad Karp and H.T Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *6th International Conference on Mobile Computing and Networking (ACM MOBICOM'00)*, pages 186–194, August 2000.

[21] Y. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–75, November 1998.

[22] Y.-B. Ko and N. H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMSCA*, 1999.

[23] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *6th International Conference on Mobile Computing and Networking (ACM MOBICOM 2000)*, pages 66–84, August 2000.

[24] B. Liang and Z.J. Haas. Predictive distance-based mobility management for PCS networks. In *Proceedings of IEEE Infocom Conference*, pages 1377–1384, 1999.

[25] C. Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2001.

[26] Charles Perkins and Elizabeth Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb 1999.

[27] I. Stojmenovic. A routing strategy and quorum based location update scheme for ad hoc wireless networks. Technical Report TR-99-09, University of Ottawa, 1999.

[28] William Su. *Mobility Prediction in Mobile/Wireless Networks*. PhD thesis, University of California, LoS Angeles, 2000.

[29] Seung-Chul M. Woo and Suresh Singh. Scalable routing protocol for ad hoc networks. *ACM/Kluwer Wireless Networks (WINET) Journal*, 7(5):513–529, 2001.