Mobile Netw Appl (2009) 14:625–637
DOI 10.1007/s11036-008-0144-3

Hash-Based Virtual Hierarchies for Scalable Location Service in Mobile Ad-hoc Networks

Wei Wang · Chinya V. Ravishankar

Published online: 17 January 2009

Abstract A location service is an essential prerequisite for geographic routing protocols for MANETs. We present VHLS, a new distributed location service protocol, that features a dynamic location server selection mechanism and adapts to network traffic workload, minimizing the overall location service overhead. We demonstrate that the ratio of location queries to updates is an important performance parameter in such protocols. Our analysis and simulations show that VHLS provides better query success rates, location service quality, and geographic routing performance than the GLS and GHLS protocols. VHLS also scales well as the network size and traffic workload increases.

Keywords mobile ad-hoc networks · location service · hashing · hierarchies

1 Introduction

Mobile ad-hoc networks (MANETs) have great potential in numerous applications, such as location tracking and retrieval [1], military operations [2], and civilian outdoor applications [3]. Nodes in MANETs are typically battery-powered devices with limited computation and communication capabilities. Messages between nodes beyond each other's radio range are routed via relaying nodes.

Routing algorithms that work well for static networks fail for MANETs, since network topology changes frequently and unpredictably. Routing in MANETs can not rely on fixed servers, but can be accomplished using either topology-based protocols [4–8], which rely on discovering and maintaining global state, or location-based (geographic) protocols, which route packets to the destination's geographic location. Geographic routing is more scalable to large MANETs, since it uses only knowledge of the destination location and local geography, and is independent of network topology and size. For a good survey of geographic routing, see [9]. Examples of geographic routing protocols include DREAM [10], LAR [11], GPSR [12], SLURP [13], and GRID [1].

Geographic routing, however, requires a sender to know the recipient's location. While nodes can obtain their own locations easily using GPS [14] or other localization hardware, locating other nodes is more difficult. Location servers must themselves be located, and messages routed to them. Besides, sending and receiving of messages consumes significant energy [15, 16]. Four desirable characteristics for location services in MANETs are listed in [1]: Service load should be well-distributed to avoid bottlenecks, node failures should have limited effect, queries for local nodes should only involve local communication, and the overhead should grow slowly with network size.

Work exists on the location service problem [1, 10, 11, 17–22]. Some methods use flooding, and are expensive, but hash-based approaches [1, 17, 21, 22] simplify the identification and use of location servers, and are far cheaper.

W. Wang · C. V. Ravishankar (✉)
Computer Science & Engineering Department,
The University of California, Riverside, Riverside, USA
e-mail: ravi@cs.ucr.edu

W. Wang
e-mail: wangw@cs.ucr.edu

1.1 Motivation and summary of our work

Most earlier work has tended to focus on improving the efficiency of updates, since the update rate is the obvious parameter that rises with mobility. In contrast, our work is motivated by the observation that while it is important to handle high update rates, a high *query* rate can actually affect performance more. Updates represent one-way traffic, but queries require two-way traffic, so higher query-update ratios can have a higher impact on servers loads, as well as on traffic in the system. We show that the ratio of location queries to location updates is an important performance parameter that has been ignored by current location services for MANETs.

We also present VHLS, a new location service protocol that is able to explicitly adapt to query-to-update ratios. VHLS manages locations using Hash-based Virtual Hierarchies [23], which have been shown to be useful in other contexts. Nodes are first organized into a hierarchy of regions, which serves as a foundation for each sending node to construct a virtual hierarchy for any destination node, using purely local information. Location updates and queries use geographic forwarding.

We compare VHLS with the Geographic Hashing Location Service (GHLS) [21] and the Grid Location Service (GLS) [1], two current hash-based location service protocols. These protocols forward all location updates and queries for a given node to a fixed number of location servers in the network, which can cause congestion under high workload. In contrast, VHLS can select a number of location servers for each node, based on the workload and the query/update ratio. A hash-based virtual hierarchy for each node defines a set of location server candidates. For a light-workload scenario, VHLS selects fewer location servers, so that location update overhead can be reduced without compromising the quality of location service. For heavier workloads, VHLS may select more location servers so that queries can be answered closer to the initiators, reducing query overhead. Our aim is to select an optimal set of location servers for each mobile node, such that the total location service overhead introduced is minimized, reducing network congestion, improving location service quality, and saving power.

In Section 2, we discuss related work. We present VHLS in Section 3, and an analytical estimate of the location service overhead of VHLS in Section 4. Simulations and performance comparisons appear in Section 5, and Section 6 concludes our paper.

2 Related work

Location service protocols in MANETs have been well-studied in recent years. The work in [21] categorizes the existing location service protocols into two taxonomies: flooding-based [10, 11] and rendezvous-based approaches. Flooding approaches are equivalent to a broad search in the network, so rendezvous-based location services are inherently more scalable [21]. Rendezvous-based methods can be subdivided into quorum-based [18, 20] and hash-based [1, 17, 21, 22] approaches. Quorum-based services associate each node n with a set of nodes, called its update quorum, to which n's location updates are sent. A location query for that node will be forwarded to a different quorum, which overlaps with the update quorum at some node, and is answered there. The work in [21] compares hash-based location services with a representative quorum-based service [20] and shows that hash-based location services are more scalable.

Many hash-based LS protocols [17, 22] are designed with particular goals. [17] proposes a hierarchical addressing model, and identifies location servers assuming that they are distributed evenly at a certain level of the hierarchy. The work in [22] presents a hierarchical approach designed specially to support communication between nodes close to each other. Hash-based methods are the most efficient reported in the literature, but differ in the way they use hashing. VHLS is most similar to GLS [1] and GHLS [21].

2.1 The grid location service (GLS)

GLS [1] uses hashing to map a node name into an integer called the node ID. The region is partitioned into a grid (see Fig. 1) and a hierarchy defined over squares of increasing size. The smallest square is referred to as an

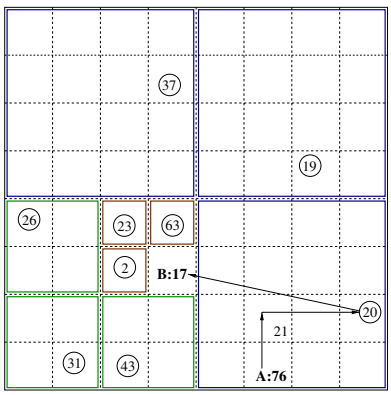


Fig. 1 GLS location servers for node B

order-1 square. Four order-i squares (sibling squares) form an order-$(i+1)$ square. GLS uses 2-hop beacons, which include each node's location and those of its neighbors.

At each level of the grid hierarchy, a node B designates as its location server the node in each of its sibling squares whose ID is closest to its own. Given N cells, the grid hierarchy has $\log_4 N$ levels, so each node designates $3 \log_4 N$ location servers. Figure 1 shows B's location servers as circles.

When A must send data to B (via geographic forwarding), it issues a location query for B. In GLS, each node stores the IDs and locations of all nodes for which it functions as a location server. At each query step, the location query is forwarded to the node with ID closest to B, until it reaches one of B's location servers. In Fig. 1, a location query from A is sent to node 21, whose ID is closest to B (17) among A's stored IDs. The query finally arrives at node 20, one of B's location servers.

B updates its location servers whenever it moves more than a threshold distance d. Servers closer to B are updated more frequently than those farther away. A node updates its order-i servers after each movement of $2^{i-2}d$ distance units [1]. B sends an update packet using geographic forwarding to the square in which the location server is located.

In GLS, B's location servers are densely distributed near B, achieving good locality. Moreover, all nodes in GLS serve as the location servers for the same number of nodes in the network, so loads are balanced. However, GLS is best for relatively static networks, since the overhead of handling node movement is very high. Once a node S moves from a region g_1 to another region g_2, it leaves a forwarding pointer at g_1. Other nodes arriving in g_1 with out-of-date location information for S can follow these pointers. This node-to-node chain may become long, greatly increasing overhead. Also, if the chain is broken, all further queries fail.

2.2 The geographic hashing location service (GHLS)

While GLS hashes node names to IDs, GHLS [21] uses geographic hashing [24] to map a node name B to a geographic *location* $H(B)$. It uses a single location server for each node; the node closest to location $H(B)$ is always the server for B.

When a location update or query arises for node B, the hash value $H(B)$ is computed, and the update or the query is forwarded using geographic routing to the node S that is closest to $H(B)$. Handoffs can be common, since S must hand off B's location information to a node S', if S' moves closer to $H(B)$ than S.

Since GHLS mandates a single location server for each node, location updates are cheaper than in GLS. Location servers are selected based on location, so GHLS handles node movement better than GLS. However, locality can be poor, since node A may be close to B, yet be very far from $H(B)$. GHLS therefore places all location servers in a region at the network's center, but this leads to uneven loads.

Figure 2 shows how GHLS selects location servers and handles location update and query. Point H represents location $H(B)$. Node D serves as B's location server since it is closer to H than any other node in the server (α) region (the inner square). Node A's query is sent to H, and therefore routed to D. B's location updates will also be routed to node D.

2.3 Other related work

Although GHLS and GLS are most directly related to our work, there is a significant amount of other, albeit less related work. Work related to routing and mobility appears in [25–28]. Various tradeoffs between complexity, robustness, and overhead appear in [1, 9, 29–31]. In [32], a performance comparison is made between GLS, GHLS, and XYLS, which is a quorum-based protocol based on the work in [20].

DREAM [10] requires nodes to maintain a database of the best known locations for all other nodes. This information is disseminated to neighboring nodes using flooding. The scope of the flooding depends on the distance traveled by the node since the last flooding operation. Routing is based on restricted directional flooding, using the direction in which the destination is expected to be located. The work in [33] is similar in spirit. Each node maintains a database of its

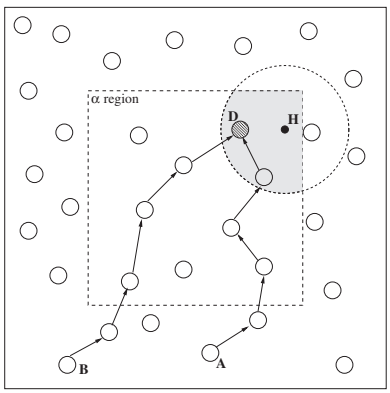


Fig. 2 Location update and query in GHLS

encounters with other nodes. Packets are routed to a destination using a *Last Encounter Routing* algorithm, which consults these databases to improve the estimate of destination's current location.

3 The VHLS protocol

We have seen how GLS hashes names into *node IDs*, and GHLS hashes them to *locations*. In contrast, VHLS hashes names first into *grid cells*, and then into servers therein. It uses a GLS-like grid hierarchy, but selects location servers very differently, using Virtual Hierarchies [23] and the Highest Random Weight (HRW) [34] hashing method. The grid hierarchy is static, but location servers are computed at query or update time. Nodes know their own locations. As GLS and GHLS do, we route all messages using geographic forwarding.

GLS and GHLS use a fixed number of location servers for each node, but VHLS selects location servers dynamically, based on load parameters such as location update rate and location query rate. We want to maximize query success rates, locality, load balancing, and minimize transmissions.

3.1 Hash-based virtual hierarchies

Hash-based virtual hierarchies are based on the HRW hashing protocol [34], and were introduced in [23], where they were applied to web caching. We apply the virtual hierarchy concept for an entirely different design goal. VHLS starts with a static hierarchy of cells as GLS [1] does, but builds dynamic and query-specific *virtual hierarchies* on it using the HRW hashing method.

Given an object with name N and a set of bucket names $b_1, b_2, \ldots, b_n$, HRW allows clients to determine, without a directory, which bucket should hold object N. It uses a hash function H to compute bucket *weights* $\{w_1^N, w_2^N, \ldots, w_n^N\}$, where $w_i^N = H(b_i, N)$. It then assigns the object to bucket b_k yielding the highest weight, that is, such that $w_k^N = \max\{w_1^N, w_2^N, \ldots, w_n^N\}$. Since H and the bucket names b_i are well-known, each client will independently compute identical object-bucket assignments.

3.1.1 Object-specific virtual hierarchies

Hierarchies often have poor performance since their upper levels aggregate requests from lower levels, causing bottlenecks [23]. We avoid this problem by creating object-specific virtual hierarchies on top of a given physical hierarchy using HRW. We use the grid hierarchy as a *skeleton* to which we apply HRW.

This skeleton is obtained by partitioning the entire network region recursively into four cells at each step, as in Fig. 3, until we reach the smallest cell (a *unit cell*), whose edge length is the minimum usable radio transmission range. Nodes in a unit cell use the two-hop distance vector protocol of GLS, so they know each other's locations.

We define *levels* for the skeleton in a bottom up fashion. The unit cell are level-1 cells, four of which form a level-2 cell, and so on. Let G_r^l denote a cell r at level l of the skeleton. Figure 3 shows a 3-level skeleton. G_1^1–G_{16}^1 are level-1 cells, G_1^2–G_4^2 are level-2 cells, and the network G is a level-3 cell.

Figure 4a represents the skeleton as a tree. We construct a virtual hierarchy for a name O_k as follows. We first apply a hashing function $\mathcal{H}$ at the root of the skeleton, and choose one of G's children, which we call the *prime cell* for G at the highest level of the skeleton. We then descend into each subregion in the second highest level of the skeleton, and apply $\mathcal{H}$ to each child of the subregion, selecting one child, which becomes the prime for the name O_k in that subregion. This procedure continues down the skeleton, until we reach the level-1 cells. The set of primes forms the virtual hierarchy.

Figure 4a shows the construction of the virtual hierarchy. The root prime is obtained by descending the hierarchy, choosing between siblings using HRW. At the top level, we have only one choice. At level 2, we apply HRW to $G_1^2, \ldots, G_4^2$, getting G_3^2 as prime. We now apply HRW across G_3^2's children $G_9^1, \ldots, G_{12}^1$, and obtain G_{10}^1, which becomes the root prime. The primes within the level-2 cells are, respectively, $\mathcal{H}(G_1^2, O_k) = G_2^1$, $\mathcal{H}(G_2^2, O_k) = G_7^1$, $\mathcal{H}(G_3^2, O_k) = G_{10}^1$, and $\mathcal{H}(G_4^2, O_k) = G_{15}^1$. Within level-1 cells, we will apply HRW over the

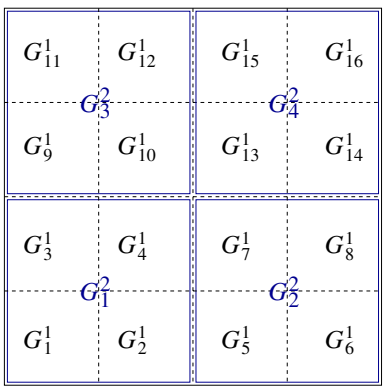
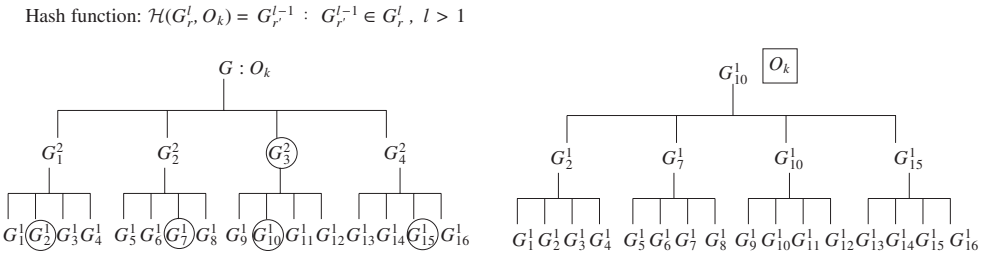


Fig. 3 Skeleton



(a) Hashing on the skeleton (b) The virtual hierarchy for object O_k

Fig. 4 Construction of the virtual hierarchy for object O_k

nodes. Figure 4b shows the resulting virtual hierarchy. G_{10}^1 is the root, and G_2^1, G_7^1, and G_{15}^1 are G_{10}^1's children.

3.1.2 Advantages of virtual hierarchies

The upper levels of typical hierarchies aggregate requests from lower levels and suffer serious congestion [23]. Such congestion is completely absent in virtual hierarchies, which are object- and request-specific. Overhead is also minimal. The skeleton and the hash function are well-known, so each node constructs the virtual hierarchy locally. Good hash functions map with equal probability to all cells, balancing workloads.

Figure 5 shows a different view of the virtual hierarchy for O_k. The root of the virtual hierarchy is G_{10}^1, where the black point resides. It points to its children G_2^1, G_7^1, and G_{15}^1, in level 2 of the virtual hierarchy, all represented by the gray points. Note that G_{10}^1 is its own child.

3.2 VHLS operation

VHLS treats grid cells as "buckets", and node names as the "object names" in Section 3.1's terminology. B builds the virtual hierarchy for its own name, and selects a set of "prime" grid cells in this virtual hierarchy as host B's location servers.

If B uses only one location server, it is located in the region defined by the virtual hierarchy's root. In other cases, B designates location servers only in the top m levels of its virtual hierarchy, choosing m to trade off update and query overheads. Update overhead grows with the number of servers, but query overhead drops, since more queries are answered locally. For higher query/update ratios, we expect greater benefit from pushing to primes farther away from the root, to cover a larger fraction of the region. We analyze how to choose m in Section 4.

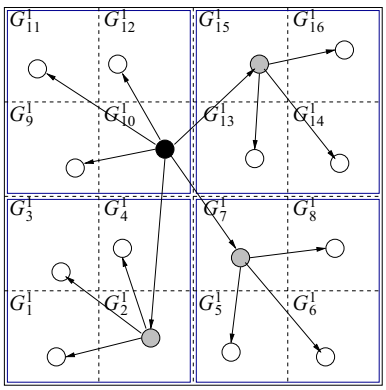


Fig. 5 Virtual hierarchy

3.2.1 Picking location servers within prime regions

We select nodes within prime regions to serve as location servers as follows. We apply HRW over the nodes $n_1^k, n_2^k, \ldots, n_m^k$ present within each prime cell c_k, selecting the node $n_{B_1}^k$ that ranks highest under HRW as B's location server in c_k. If n_{B_1} is unavailable due to movement or failure, we select n_{B_2}, the next highest-ranking node.

3.3 Handling updates and queries

B sends updates and m's value to its prime cells in the top m levels using geographic forwarding. A server in a prime cell at level l stores this update, forwarding it iff $l < m$. All location services must manage server caches using a suitable policy. TTLs based on a node's *update interval* are a reasonable choice.

A querying node first computes B's virtual hierarchy, and queries B's prime node p_B in the local level-1 cell, the local "authority" for B's location. If p_B does not have B's location, it computes B's virtual hierarchy and forwards the query to the prime of the enclosing level-2 cell. In Fig. 5, a query in cell G_{16}^1 would go first to the node in G_{16}^1 with the maximum HRW value. If this server did not have the value, it would recursively query the prime for the enclosing level-2 cell (cell G_{15}^1, the gray node), etc.

3.4 Handling location server mobility

Queries and updates are forwarded to prime cells. Hence, when a node moves out of a cell, the location information in it must be kept in the cell. VHLS solves this problem elegantly. Let a prime cell contain nodes $n_1, n_2, \ldots, n_k$, and let HRW induce the order $n_{i1}, n_{i2}, \ldots, n_{ik}$ for B's location. That is, $\mathcal{H}(n_{i1}, B) > \mathcal{H}(n_{i2}, B) > \cdots > \mathcal{H}(n_{ik}, B)$. VHLS will select node n_{i1} to hold B's location information.

3.4.1 Join/leave protocol

Before n_{i1} leaves the cell, it simply transfers the data it holds for B to n_{i2}, which is second in the HRW ordering for B in the cell. If a query for B's location now arrives, HRW automatically selects n_{i2}.

Joins raise a different issue. When a new node n_j enters a cell, it may turn out that $\mathcal{H}(n_j, B) > \mathcal{H}(n_{i1}, B)$, so that HRW would now select n_j, instead of n_{i1}. Upon join, n_j broadcasts a beacon containing its ID and its location to all nodes in this cell. Each node in the cell uses HRW to check if any of the location data it holds should be re-mapped to n_j, and passes such information to n_j.

Our method has low overhead, since beacons are standard in MANETs, and broadcast only within the cell. Data transfer occurs only when n_j ranks at the top of the HRW list for data present in the cell. Our experiments show that handoff overhead is much lower in VHLS than in GHLS.

3.5 Handling node failure

Location server failures can have serious consequences for location-based routing, but VHLS handles failures very cleanly. Within a cell, updates and queries for a node B are sent to n_{B_1}, the node that ranks highest under HRW. When such a message times out, due to node failure or mobility, the next-ranking node n_{B_2} is selected as B's new location server (see Section 3.2.1). All further updates and queries for node B will be sent to n_{B_2}. When n_{B_1} becomes available again, it simply executes the join protocol described in Section 3.4.1.

3.6 Some enhancements

3.6.1 Server overloads

VHLS handles hot-spots, transient overloads, and query bursts exactly as it handles node failures. Congested nodes can simply refuse to respond for some time, during which location updates and queries are redirected to the next node in the ordering defined by HRW. Congested servers may reactivate themselves whenever they wish.

3.6.2 Location query forwarding

Instead of forwarding queries up the hierarchy of primes, a relaying node can compute its Euclidean distances to the next prime as well as to the root prime, and send the query directly to the root prime if it is closer. A query response is guaranteed at the root prime, so this optimization reduces prime-to-prime forwarding overhead.

3.6.3 Location update snooping

Location update messages can be overheard by other nodes. A node that overhears a location update for node B can determine if it is the prime in its own cell for B, and store the update. It can respond to future location queries for B without forwarding.

3.6.4 Spanning trees over primes

All nodes get the skeleton at start-up, so they can compute the level-l prime cells for any value of l. A node can construct a spanning tree that covers exactly these prime cells, and disseminate updates over this spanning tree, using geographic routing.

4 Analytical estimates of VHLS costs

Let us label the hierarchy in bottom-up fashion. The leaf-level nodes are at level 1, and the root is at level L, if the depth of the skeleton is L. All nodes, except leaves, have fan-out 4, since each grid cell has four children.

We estimate the location service overhead based on the following assumptions. A location update is disseminated to its servers along the edges of the spanning tree constructed to cover the regions in which the servers reside. If node A wants to communicate with node B, it will initiate a query for B's location, which is sent up the virtual hierarchy towards the root. When the query reaches one of B's location servers, a reply is sent back to A directly. All location updates, queries and query replies are forwarded using geographic forwarding.

As in [21], we will estimate the number of hops to send a message over a distance d to be d/r, if the radio communication range is r. It is known [35] that the

expected distance between two random points in the interior of a unit square is

$$\delta = \frac{1}{15}\left[\sqrt{2} + 2 + 5\ln\left(1+\sqrt{2}\right)\right] = 0.521405433\cdots \quad (1)$$

For a $s \times s$ square, this expectation is hence $s\delta$. This value serves a lower bound on the length of a route between these points, improving as an approximation to it with increasing node density. Level-1 VHLS regions are "unit" squares, having a side length equal to the lower bound on the radio communication range. Hence the expected number of hops per message in a level-i square, which is of size $2^{i-1} \times 2^{i-1}$, will be

$$h_i = 2^{i-1}\delta. \quad (2)$$

We want to send location updates for node B to its servers in the prime cells in the top m levels of B's virtual hierarchy, and let location queries for B proceed up the virtual hierarchy. We choose m to minimize the number of data transmissions.

Let an *update interval* be the time a node takes to move a threshold distance, so that a location update is sent each update interval. At level m, let $C_U(m)$ be the number of data transmissions required for one update, $C_Q(m)$ be the number of data transmissions for all queries initiated in this interval, and $C_R(m)$ be the number of data transmissions for query responses in this interval. Hence the total transmission cost per update interval is $C(m) = C_U(m) + C_Q(m) + C_R(m)$. To keep the analysis tractable, we assume as [21] does, that the nodes are static and uniformly distributed, without network interference, data retransmissions, or caching. Instead, we demonstrate through experiments that our methods are robust in the face of these effects.

4.1 Estimating traffic and costs

Let the root be at level L, and assume updates for B's location have been pushed down to level-m regions, so that there are 4^{L-m} location servers holding B's location. Let λ queries for B's location arise uniformly among the nodes in the network in one update interval. Each query first goes to the prime node in its level-1 region (see Section 3.3), and is answered at unit cost if this region has one of B's 4^{L-m} servers. The number of queries answered locally is therefore

$$R_1 = \frac{4^{L-m}}{4^{L-1}}\lambda = \frac{1}{4^{m-1}}\lambda,$$

and the number of queries forwarded to level-2 primes is

$$Q_1 = \left(1 - \frac{1}{4^{m-1}}\right)\lambda.$$

There are 4^{L-2} level-2 regions, and a lower bound on the expected cost per message at this level is 2δ (see Eq. 2). As before, the expected number of queries answered at level 2 is

$$R_2 = \frac{4^{L-m}}{4^{L-2}}Q_1 = \frac{1}{4^{m-2}}\left(1 - \frac{1}{4^{m-1}}\right)\lambda,$$

and the number of queries forwarded to level-3 primes is

$$Q_2 = \left(1 - \frac{1}{4^{m-2}}\right)\left(1 - \frac{1}{4^{m-1}}\right)\lambda.$$

Propagating the query at level 3 has cost bounded below by $2^2\delta$. Proceeding thus, we obtain a lower bound on the query cost per update interval as

$$\begin{aligned} C_Q(m) &= \lambda + 2\delta \cdot Q_1 + 2^2\delta \cdot Q_2 + \cdots \\ &= \lambda + \lambda\delta \sum_{k=1}^{m-1}\left[2^k \prod_{i=1}^{k}\left(1 - \frac{1}{4^{m-i}}\right)\right]. \end{aligned}$$

Similarly, the query reply costs per update interval are

$$\begin{aligned} C_R(m) &= R_1 + 2\delta \cdot R_2 + \cdots \\ &= \frac{1}{4^{m-1}}\lambda + \lambda\delta \sum_{k=1}^{m-1}\left[\frac{2^k}{4^{m-k}} \cdot \prod_{i=1}^{k}\left(1 - \frac{1}{4^{m-i}}\right)\right]. \end{aligned}$$

To estimate update cost, we note that B sends one location update per update interval to each of 4^{L-m} location servers, which are uniformly distributed over the network. Updates can be sent over a minimum spanning tree built over the cells holding these servers. It is known [36] that the minimum spanning tree over n random points in a unit square has an expected size of $0.656\sqrt{n}$. Node B has 4^{L-m} level-m primes, in a $2^{L-1} \times 2^{L-1}$ region. Hence the cost of sending updates over the spanning tree is bounded below by $0.656 \times 2^{L-1}\sqrt{4^{L-m}} = 0.656 \times 2^{2L-(m+1)}$.

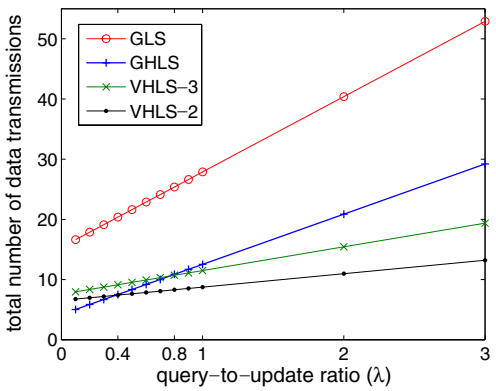


Fig. 6 Costs per update interval compared

Table 1 Corrected analysis for GLS and GHLS

Metrics	GLS	GHLS
Update cost	$\frac{3}{2} \cdot (2+\sqrt{2}) \cdot (L-1)$	$0.5214 \cdot 2^{L-1}$
Query cost	$0.5214 \cdot 2^{L}$	$0.5214 \cdot 2^{L-1}$
Reply cost	$0.5214 \cdot 2^{L-1}$	$0.5214 \cdot 2^{L-1}$

Node B reaches this spanning tree by sending this update to the closest location server. For n random points in unit area, the distance to the closest point [37] has the density $f(x) = 2\pi nxe^{-\pi nx^2}$. The expected distance to the closest point is

$$\int_0^\infty 2\pi nx^2 e^{-\pi nx^2} dx = \sqrt{\frac{1}{4n}}.$$

The spanning tree has 4^{L-m} location servers in a $2^{L-1} \times 2^{L-1}$ square, so this expectation is $2^{L-1}\sqrt{\frac{1}{4^{L-m+1}}} = 2^{m-2}$. Hence,

$$C_U(m) = 0.656 \times 2^{2L-(m+1)} + 2^{m-2}$$

We push updates to the optimal level m^* which minimizes the cost $C(m^*) = C_U(m^*) + C_Q(m^*) + C_R(m^*)$. A grid with N level-1 cells has no more than $\lceil \log_4 N \rceil$ levels, so m^* is efficiently found by searching over all values $0 \le m^* \le L$. Experimentally, we found $m^* = 3$ for query/update ratios below 4, and $m^* = 2$ above 4 (Figs. 12, 13, 14, and 15).

Figure 6 shows analytical results for VHLS, GLS, and GHLS for a 4-tier hierarchy. Experimental comparisons are in Section 5. The value $\delta = \frac{\sqrt{2}}{3}$ is incorrectly used in [21] for the average distance between two random points in a unit square. Table 1 shows a revised analysis using the correct value (Eq. 1). The metrics for GLS and GHLS in the table correspond to our $C_U(m)$, $C_Q(m)$, and $C_R(m)$ respectively. GHLS has low costs for low query/update ratios. However, as this ratio increases, VHLS performs best. Updates are pushed to level 3 in VHLS-3 and to level 2 in VHLS-2, which works better for higher query/update ratios, as expected.

5 Experimental evaluation

Our extensive simulations under *ns2* [38] show that VHLS has far lower protocol overhead than GLS and GHLS, yet performs significantly better under standard location services metrics. We obtained a GLS implementation from its authors, but re-implemented GHLS and VHLS. All used the GPSR [12] geographic forwarding protocol. For a fair comparison of VHLS with GLS and GHLS, we used simulation scenarios similar to those in [1] and [21]. As in [1] and [21], we used the random waypoint model [39], although other mobility models [40–42] may be more realistic. A node chooses a random destination and moves towards it with a constant speed picked randomly between zero and some maximum. It may pause briefly at the destination before moving again.

Nodes use the IEEE 802.11 radio and MAC model (CMU extension). As in [1], radios used a 1 Mbps rate when no data traffic was present, and a 2 Mbps rate otherwise. For each run, we put the ns2 simulator in "cold start" mode for 50 simulated seconds to eliminate transient effects, and then ran each simulation for 300 simulated seconds. We computed results as the average over 5 simulation runs. The duration and number of simulations are identical to those used in [1, 21] for evaluating the performance of GLS and GHLS. The cold start interval was effective in smoothing out transients and reducing variance across runs; the standard deviation/mean ratio was well below 1% in our experiments. We therefore report means only, to avoid cluttering the graphs.

Table 2 summarizes our experimental parameters. The region was a 2 km × 2 km square area partitioned into 64 unit regions, forming a 4-level skeleton with 250 m × 250 m unit regions. There were 100 nodes per square kilometer on average, the node density chosen in [1] for a system to be used over relatively large areas such as a campus or a city. The maximum node speed was 10 m/s. Each node in the network initiates 20 queries for the locations of randomly chosen nodes between 10 and 300 s in each simulation run.

Table 2 Simulation parameter settings

Parameters	Default	Parameters	Default
MAC protocol	802.11	Region size	2 km × 2 km
Radio model	TwoRayGround	Unit region size	250 m × 250 m
Radio range	250 m	Hierarchy height	4
Routing protocol	GPSR	Node density	100 km^2
Mobility model	Random waypoint	# nodes in network	400
Mobility pause time	0	Max. node speed	10 m/s
Cold start time	50 s	Update threshold	200 m
Simulation time	300 s	Queries per node	20

The update distance threshold is fixed at 200 m. The pause time between two random waypoint movements is 0.

5.1 Scalability and location service protocol overhead

We first evaluated the inherent overheads of GLS, GHLS, and VHLS by having each node initiate queries for randomly chosen locations. Traffic comprised only MAC beacons and location service (LS) protocol packets, which included location update packets, location query and reply packets, as well as handoff packets in GHLS and VHLS. Figure 7 shows the absolute LS protocol overheads (left Y-axis, dashed lines) and the relative ratios of the protocol overheads to that of VHLS (right Y-axis, solid lines), for varying region sizes having 100 nodes/km^2. Each node issued 20 queries for randomly chosen nodes. As region size grew from 4 km^2 to 16 km^2, VHLS had the most modest overhead growth. The GLS/VHLS overhead ratio grew from 3 to as much as 7. The corresponding ratio for GHLS grew from 2 to 3. VHLS clearly has better scalability.

Figure 8 shows how absolute (left Y-axis, dashed lines) and relative (right Y-axis, solid lines) LS protocol overheads grow for query-to-update ratios up to 4, under conditions described in Section 5.3. VHLS overhead growth is clearly modest. GHLS overhead grows significantly faster, becoming as high as 2.4 times that of VHLS. GLS overhead grows most dramatically, reaching nearly 6 times that of VHLS.

Further evidence of the scalability of VHLS is in Fig. 14, which compares the overhead of geographic routing under the three methods. As query/update ratio rises, overhead barely grows under VHLS, but grows robustly under GLS and GHLS.

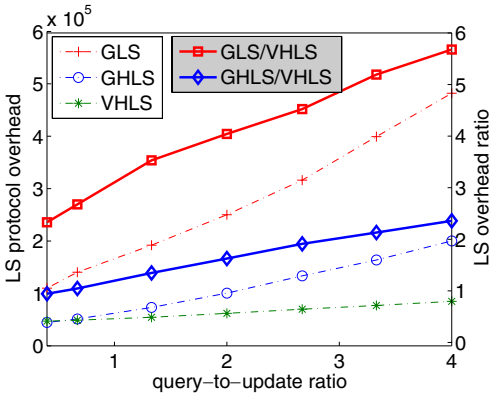


Fig. 8 Effects of network workload on LS protocol overhead

5.2 Query success performance

We now show that VHLS outperforms GLS and GHLS under standard location service performance metrics, starting with the *Query Success Ratio*. A query is successful if its initiator receives a reply. As in [1] and [21], we do not retransmit failed queries, and measure the fraction of successful queries.

Figure 9 shows the effects of speed on Query Success Ratio (QSR) for the three schemes in a network of 400 nodes. The maximum node speeds were set at 10, 30, and 50 m/s, and each node randomly picked a speed from the given range. Other parameters were as in Table 2. QSR decreases with speed for all three schemes because more location servers will contain out-of-date location information. Faster nodes also generate more updates, increasing traffic. VHLS is least sensitive to node speed because it minimizes the total location service overhead for each update (see Section 4), causing the least network congestion and packet loss.

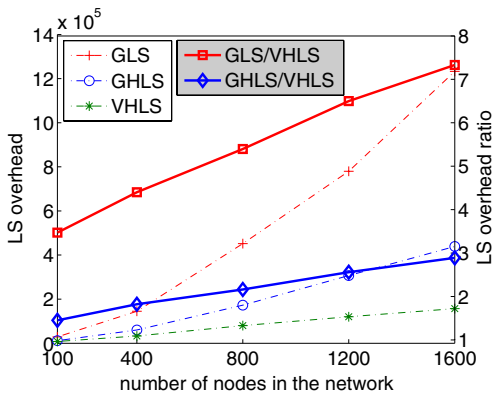


Fig. 7 Scalability as a function of network size

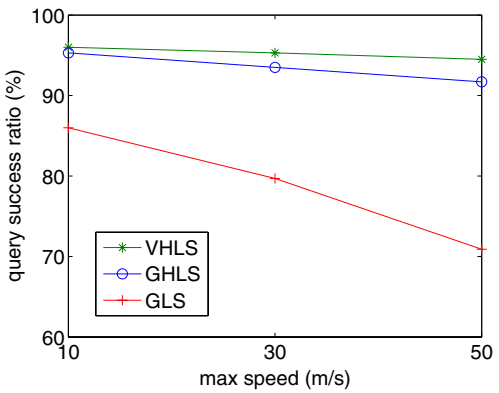


Fig. 9 QSR as a function of node max speed (m/s)

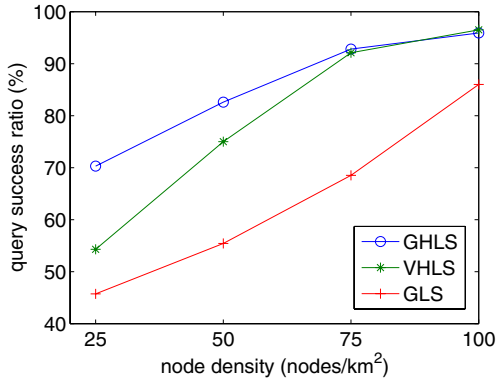


Fig. 10 Effects of node density on QSR

Figure 10 shows the effect of node density on QSR for the three schemes. VHLS has lower QSR than GHLS at low node densities, which result in sparser unit regions. A node density of 25 nodes/km^2 results in about 1.56 nodes on average in each unit region, so that prime cells may sometimes be empty, causing location update or query failure. However, as the node density increases, the QSR of VHLS rises quickly. Beyond a node density of 75 nodes/km^2, VHLS and GHLS have similar QSR. When node densities are low, we can change the algorithm of Section 3.2.1 to include nodes from cells neighboring the chosen prime cell, when picking servers.

Figure 11 shows the effects on QSR of the query rate. As expected, QSR drops with query load for all three schemes, but VHLS outperforms GLS and GHLS. Moreover, as Fig. 6 shows, GHLS has greater message overhead than VHLS when the query load increases. VHLS is clearly the best method for high query loads.

5.3 Ability to support message delivery

Let node B be at location (x_B, y_B), and let a query for B's location return (x'_B, y'_B). The query reply is *sufficiently accurate* for packet delivery if (x'_B, y'_B) and (x_B, y_B) are within radio range of each other. We evaluated the quality of location service protocols in terms of the fraction of queries returning sufficiently accurate responses, using packet delivery ratio as a metric. Each node issued a location query for a random node, sending it packets after receiving a response.

We use the *query-to-update ratio* to characterize the workload. Each node in the network moved at a speed between 0 and 10 m/s, giving an average speed of 5 m/s. For an update distance threshold of 200 m, the time between updates is 40 s on average, corresponding to 7.5 updates per node in each simulation run. Each node generates 15 data packets per run, choosing a random destination for each, resulting in about 15 location queries per node, giving a query-to-update ratio of 2.

Figure 12 shows effects of the query/update ratio on QSR. We do not retransmit failed queries. Clearly, VHLS degrades least in performance, and GLS degrades most as the query/update ratio grows, an effect easily understood from Fig. 8. The number of location service protocol packets for GLS increases rapidly with network workload. The presence of data packets increases network traffic, further diminishing QSR. The advantage of VHLS over GHLS becomes greater as the query/update ratio increases.

Figure 13 shows that the packet delivery ratio is close to the query success ratio for all three schemes for low network workloads, so that a successful query indicates a high probability of data packet delivery. However, as network traffic increases, GLS and GHLS incur significant protocol overhead, causing greater network

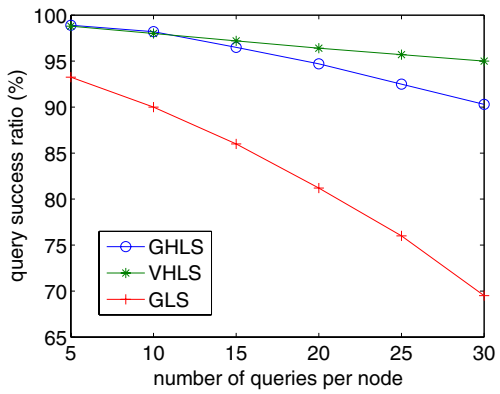


Fig. 11 Effects of number of queries per node on QSR

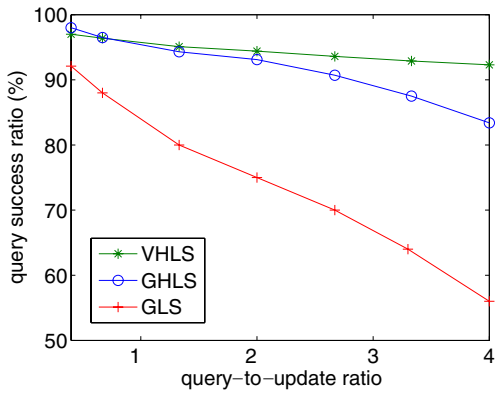


Fig. 12 Effects of network workload on QSR

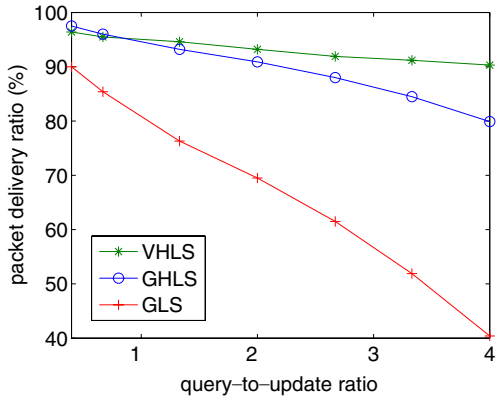


Fig. 13 Effects of workload on packet delivery ratio

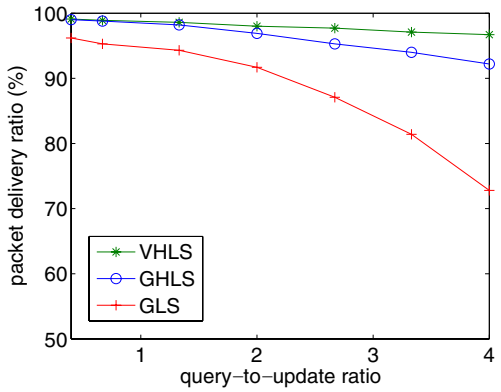


Fig. 15 Packet delivery ratio for geographic routing

congestion and packet loss. VHLS outperforms GLS and GHLS for heavy network workloads.

5.4 Support for geographic routing

We studied how well these each of these schemes supports geographic routing, and included data traffic to each source-to-destination connection. In each run, every node in the network originated 3, 5, 10, 15, and 20 queries, respectively, to a random location. Upon receiving a response, the source sent 4 data packets per second to the same destination for a duration of 10 s. Unlike the simulations in previous sections, location queries were re-transmitted in case of query failures. Mobility parameters are as in Table 2. As in [21], we examined geographic routing performance in terms of routing overhead and packet delivery ratio (PDR) over various network workloads. Routing overhead includes LS protocol overhead and geographic beaconing overhead.

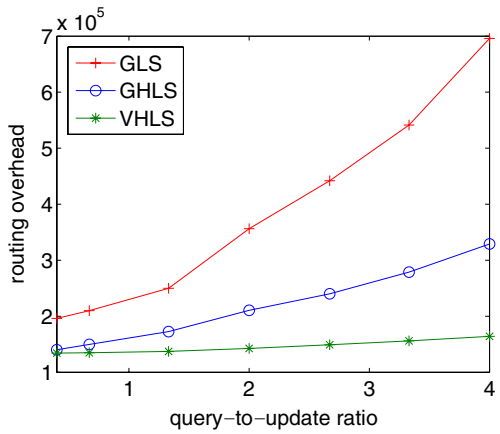


Fig. 14 Effects of network workload on routing overhead

Figure 14 shows the effects of network workload on routing overhead. For light workloads, each node initiated 3 or 5 queries per simulation run, corresponding to query-to-update ratios of 0.4 and 0.67 respectively. VHLS and GHLS have significantly lower routing overhead than GLS. The gap between VHLS and GHLS widens as the query-to-update ratio increases. VHLS clearly performs very well in heavy traffic.

Figure 15 shows the effect of network workload on data packet delivery ratio. Packet delivery failures may occur for two reasons. First, there may be query failures due to empty prime regions or location server handoff failures. Second, a query response may not be sufficiently accurate to ensure packet delivery (see Section 5.3). All protocols show high PDR for low network workloads, and VHLS and GHLS scale well as the query load increases. GLS performance worsens with traffic workload due to the quick increase in protocol overhead. This is quite consistent with the results in Fig. 14.

VHLS superficially resembles GLS more than GHLS, since it uses a grid structure. However, GHLS shows performance closer to that of VHLS than does GLS. GLS maps the node name to a node ID rather than to a grid cell, and there is no connection between the location of a node and its ID. In contrast, GHLS maps the node name to a physical location, which is closer to the VHLS strategy of hashing to a grid cell. This strategy clearly works better with geographic routing.

6 Conclusion

VHLS is a new distributed location service protocol for supporting geographic routing in wireless mobile ad-hoc networks. VHLS uses a hash-based virtual hierarchy scheme for both location updates and queries.

Unlike physical hierarchies, whose upper levels aggregate requests from lower levels, and cause bottlenecks and congestion, virtual hierarchies are node-specific, and do not suffer from this drawback. VHLS partitions the region into a regular hierarchy of subregions of increasing size, using which, a node A can compute the virtual hierarchy corresponding to any other node B. A can thus determine the locations of B's location servers using purely local information.

We have shown how VHLS has the potential to be *tunable*, selecting a number of location servers based on the network traffic and workload. We also presented a quantitative analysis of the location service overhead. Theoretically, we can find an optimal set of location servers for each node such that the overall location service overhead can be minimized. Our use of an optimal set of servers reduces network congestion, improves location service quality, and saves significant power.

We also experimentally compared VHLS with GLS and GHLS, using the network simulator ns2, taking into account realistic network issues, such as interference between wireless devices, transmission delay, and packet loss. VHLS has the lowest overhead and is the most scalable of the three. VHLS generally also outperformed the other protocols in terms of location query success ratio under various node speeds and query loads. It has better location service quality in terms of the fraction of successful queries. Finally, since location service protocols are intended to support geographic routing, we showed that VHLS is superior in this regard as well. VHLS outperforms other current methods, particularly under higher traffic conditions and query/update ratios.

VHLS is a new approach to managing location servers for MANETS, which can be tuned to work well in a range of mobility scenarios. Future work will include considering improvements to the current VHLS protocol, some of which we have already described, and exploring its performance in the context of specific applications.

Acknowledgement This work was supported in part by a grant from Tata Consultancy Services, Inc.

References

1. Li J, Jannotti J, Couto DSJD, Karger DR, Morris R (2000) A scalable location service for geographic ad hoc routing. In: MobiCom'00: proceedings of the 6th annual international conference on mobile computing and networking, pp 120–130
2. Tate A, Levine J, Jarvis P, Dalton J (2000) Using AI planning technology for army small unit operations. In: Artificial intelligence planning systems, pp 379–386
3. Morris R, Jannotti J, Kaashoek F, Li J, De Couto D (2000) Carnet: a scalable ad hoc wireless network system. In: Proceedings of the 9th ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, September
4. Perkins C, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of the ACM SIGCOMM'94 conference on communications architectures, protocols and applications, pp 234–244
5. Johnson DB, Maltz DA (1996) Dynamic source routing in ad hoc wireless networks. In: Mobile computing, vol 353. Kluwer Academic, New York
6. Perkins CE, Royer EM (1999) Ad-hoc on-demand distance vector routing. In: Proceedings of the 2nd IEEE workshop on mobile computing systems and applications, New Orleans, LA, pp 90–100, February
7. Park VD, Corson MS (1997) A highly adaptive distributed routing algorithm for mobile wireless networks. In: Proceedings of INFOCOM, pp 1405–1413
8. Haas ZJ, Pearlman MR, Samar P (2002) The zone routing protocol (ZRP) for ad hoc networks. IETF MANET Working Group. INTERNET-DRAFT, July. [Online]. Available http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt
9. Mauve M, Widmer J, Hartenstein H (2001) A survey on position-based routing in mobile ad-hoc networks. IEEE Netw Magazine 15:30–39, November–December
10. Basagni S, Chlamtac I, Syrotiuk VR, Woodwar BA (1998) A distance routing effect algorithm for mobility (DREAM). In: Proceedings of the 4th annual ACM/IEEE international conference on mobile computing and networking (MobiCom' 98), pp 76–84.
11. Ko Y-B, Vaidya NH (2000) Location-aided routing (LAR) in mobile ad hoc networks. Wirel Netw 6(4):307–321, July
12. Karp B, Kung HT (2000) GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the sixth annual international conference on mobile computing and networking (MobiCom 2000), pp 243–254
13. Woo S-CM, Singh S (2001) Scalable routing protocol for ad hoc networks. Wirel Netw 7(5):513–529, September
14. Bulusu N, Heidemann J, Estrin D (2000) Gps-less low cost outdoor localization for very small devices. IEEE Pers Commun Magazine 7(5):28–34, October
15. Stemm M, Katz RH (1997) Measuring and reducing energy consumption of network interfaces in hand-held devices. IEICE Trans Commun E80-B(8):1125–1131
16. Feeney LM, Nilsson M (2001) Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In: INFOCOM, pp 1548–1557
17. Xue Y, Li B, Nahrstedt K (2001) A scalable location management scheme in mobile ad-hoc networks. In: Proceeding of the IEEE conference on local computer networks—LCN'2001
18. Hass ZJ, Liang B (1999) Ad hoc mobility management with uniform quorum systems. IEEE/ACM Trans Netw 7(2):228–240, April
19. Cheng CT, Lemberg HL, Philip SJ, van den Berg E, Zhang T (2002) SLALoM: a scalable location management scheme for large mobile ad-hoc networks. In: Proceedings of IEEE wireless communications and networking conference (WCNC 2002), pp 574–578, March
20. Stojmenovic I, Vukojevic B (1999) A routing strategy and quorum based location update scheme for ad hoc wireless networks. Computer Science, SITE, University of Ottawa, Tech. Rep. TR99-09, September

21. Das SM, Pucha H, Hu YC (2005) Performance comparison of scalable location services for geographic ad hoc routing. In: Proceedings of IEEE INFOCOM, vol 2, pp 1228–1239, March
22. Kiess W, Fuessler H, Widmer J, Mauve M (2004) Hierarchical location service for mobile ad-hoc networks. SIGMOBILE Mob Comput Commun Rev 8(4):47–58
23. Yao Z, Ravishankar CV, Tripathi S (2001) Hash-based virtual hierarchies for caching in hybrid content-delivery networks. UCR, Tech. Rep. 62, May
24. Ratnasamy S, Karp B, Yin L, Yu F, Estrin D, Govindan R, Shenker S (2002) GHT: a geographic hash table for data-centric storage. In: Proceedings of the first ACM international workshop on wireless sensor networks and applications (WSNA'02), pp 78–87, September
25. Kuhn F, Wattenhofer R, Zollinger A (2003) Worst-case optimal and average case efficient geomentric ad hoc routing. In: 4th ACM MOBIHOC, June
26. Jain R, Puri A, Sengupta R (2001) Geographic routing using partial information for wireless ad-hoc networks. IEEE Pers Commun 8(1):48–57
27. Karp B, Kung HT (2000) Greedy perimeter stateless routing for wireless networks. In: ACM MOBICOM, August
28. Bose P, Morin P, Stojmenovic I, Urrutia J (1999) Routing with guaranteed delivery in ad-hoc wireless networks. In: 3rd intnl. workshop on discrete algorithms and methods for mobile computing and communications, August
29. Giordano S, Hamdi M (1999) Mobility management: the virtual home region. EPFL, Lausanne, Switzerland, Tech. Rep. SSC/1999/037, October
30. Hsiao PH (2002) Geographical region summary service for geographical routing. Mob Comput Commun Rev 5:25–39
31. Wong VWS, Leong VCM (2001) An adaptive distance-based location update algorithm for next-generation pcs networks. IEEE J Sel Areas Commun 19(10):1942–1952
32. Das SM, Pucha H, Hu YC (2007) On the scalability of rendezvous-based location services for geographic routing. Comput Netw 51:3593–3714
33. Grossglauser M, Vetterli M (2006) Locating mobile nodes with ease: learning efficient routes from encounter histories alone. IEEE/ACM Trans Netw 14(3):457–469, June
34. Thaler DG, Ravishankar CV (1998) Using name-based mappings to increase hit rates. IEEE/ACM Trans Netw 6(1):1–14, February
35. Ghosh B (1951) Random distances within a rectangle, and between two rectangles. Bull. Calcutta Math Soc 43:17–24
36. Steele JM (1988) Growth rates of euclidean minimal spanning trees with power weighted edges. Ann Probab 16(4):1767–1787
37. Dacey MF (1972) Distance between reflexive nearest neighbors in a poisson point process. Econ Geogr 48(2):212–213, April
38. McCanne S, Floyd S (1995) ns network simulator. http://www.isi.edu/nsname/ns/. Accessed Dec 2008
39. Broch J, Maltz DA, Johnson DB, Hu Y-C, Jetcheva J (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. In: ACM/IEEE MobiCom, pp 85–97, October
40. Bettstetter C (2001) Mobility modeling in wireless networks: categorization, smooth movement, and border effects. ACM Mob Comput Commun Rev 5(3):55–66
41. Camp T, Boleng J, Davies V (2002) A survey of mobility models for ad hoc network research. Wirel Commun Mob Comput (WCMC): special issue on mobile ad hoc networking: research, trends and applications 2(5):483–502
42. Jardosh A, Belding-Royer E, Almeroth K, Suri S (2003) Towards realistic mobility models for mobile ad hoc networks. In: ACM MobiCom, pp 217–229

Wei Wang received the bachelor's and master's degrees in Computer Science and Engineering from Northeastern University, Shenyang, China, and her Ph.D. degree in the department of Computer Science and Engineering at the University of California, Riverside. Her research interests are in the areas of distributed systems and networks, and particularly in wireless and mobile communication environments.

Chinya V. Ravishankar has been with the University of California at Riverside since Fall 1999, where he is currently Professor of Computer Science and Engineering, and Associate Dean in the Bourns College of Engineering. Between 1986 and 1999, he was on the Faculty of the Electrical Engineering and Computer Science Department at the University of Michigan—Ann Arbor. Prof. Ravishankar's recent research has been in the areas of Databases, Networking, and Security. He holds an undergraduate degree in Chemical Engineering from the Indian Institute of Technology, Bombay, and the M.S. and Ph.D. degrees in Computer Sciences from the University of Wisconsin—Madison. He is a Senior Member of the Institute of Electrical and Electronics Engineers, and a member of the Association for Computing Machinery.